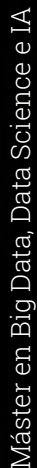


Laura Rodríguez Roperó



Modelos de Regresión Lineal y Logística

Práctica 1 de el módulo de Minería de Datos y Modelización Predictiva

por

Laura Rodríguez Roperó

06/02/2025

Índice

1. Introducción al objetivo del problema y las variables implicadas	1
2. Importación del conjunto de datos y asignación de los tipos de las variables	2
3. Análisis descriptivo del conjunto de datos	3
4. Corrección de los errores detectados	5
5. Análisis de valores atípicos	7
6. Análisis de valores perdidos	8
7. Detección de las relaciones entre las variables	10
8. Construcción del modelo de regresión lineal	14
9. Construcción del modelo de regresión logística	19

Profesor:	Rosa Espinola
Facultad:	Facultad de Estudios Estadísticos
Universidad:	Universidad Complutense de Madrid
Máster:	Big Data, Data Science e Inteligencia Artificial



Introducción al objetivo del problema y las variables implicadas

El objetivo de esta práctica es desarrollar dos modelos de regresión para analizar fenómenos relacionados con los resultados electorales en municipios de España, basándonos en una variable continua y una variable binaria. A través de estos modelos, se busca comprender mejor los factores asociados a los comportamientos electorales y la participación ciudadana en las elecciones.

El conjunto de datos incluye diversas variables explicativas que abarcan dimensiones demográficas, económicas, sociales y geográficas. Entre ellas destacan:

Demografía: Porcentajes de población según franjas de edad, porcentaje de mujeres, proporción de población extranjera y distribución geográfica. Estas variables reflejan las características sociodemográficas de los municipios.

Economía y empleo: Datos sobre tasas de desempleo por sector (agricultura, industria, servicios) y edad, así como el número total de empresas y su distribución sectorial. Estos factores pueden influir en la participación electoral debido a la relación entre estabilidad económica y compromiso político.

Infraestructura y densidad: Superficie del municipio, densidad poblacional, número de inmuebles y promedio de personas por inmueble. Estas características pueden afectar la accesibilidad a los colegios electorales y la cohesión comunitaria.

Movilidad y origen: Porcentajes de población nacida en la misma comunidad autónoma o en otras, lo cual puede relacionarse con la integración y el sentido de pertenencia política.

Teniendo todo esto en cuenta, he decidido seleccionar como variables objetivo a **AbstentionPtge** (variable continua) y **AbstencionAlta** (variable binaria). La primera refleja la participación ciudadana, un aspecto crucial en los estudios electorales, y la segunda simplifica el análisis al enfocarse en una condición concreta: si la abstención supera un umbral del 30%. Ambas están interrelacionadas, lo que facilita la comparación entre análisis detallados y simplificados.

Finalmente, la abstención es un fenómeno central que puede influir en los resultados electorales, y las variables elegidas permiten explorar tanto factores que la afectan como municipios con patrones atípicos.

Ahora bien, el análisis tiene como objetivo:

1. Identificar los factores que explican los niveles de abstención electoral mediante un modelo de regresión lineal.
2. Determinar las probabilidades de alta abstención en un municipio mediante un modelo de regresión logística.
3. Contribuir al diseño de políticas que reduzcan la abstención y fomenten la participación electoral.

2

Importación del conjunto de datos y asignación de los tipos de las variables

```
1 # Cargo los datos
2 datos = pd.read_excel('DatosEleccionesEspaña.xlsx')
3
4 # Elimino las variables objetivo descartadas
5 datos=datos.drop(['Dcha_Pct', 'Izda_Pct', 'Otros_Pct', 'Derecha', 'Izquierda'], axis=1)
6
7 # Compruebo el tipo de formato de las variables que se han asignado en la lectura.
8 datos.dtypes
9
10 # Indico las categóricas que aparecen como numéricas
11 numericasAcategoricas = ['CodigoProvincia', 'AbstencionAlta']
12
13 # Las transformo en categóricas
14 for var in numericasAcategoricas:
15     datos[var] = datos[var].astype(str)
16
17 # Genero una lista con los nombres de las variables.
18 variables = list(datos.columns)
19
20 # Selecciono las columnas numéricas y categóricas del DataFrame
21 numericas = datos.select_dtypes(include=['int', 'int32', 'int64', 'float', 'float32', 'float64']).columns
22 categoricas = [variable for variable in variables if variable not in numericas]
23
24 # Frecuencias de los valores en las variables categóricas
25 analizar_variables_categoricas(datos)
26
27 # Cuento el número de valores distintos de cada una de las variables numéricas de un
    DataFrame, y compruebo que no hay ninguna que tenga menos de 10 valores diferentes
28 cuentaDistintos(datos)
29
30 # Compruebo que todas las variables tienen el formato que quiero
31 datos.dtypes
```

Out[]: Name object, CodigoProvincia object, CCAA object, Population int64, TotalCensus int64, AbstentionPtge float64, AbstencionAlta object, Age_0-4_Ptge float64, Age_under19_Ptge float64, Age_19_65_pct float64, Age_over65_pct float64, WomanPopulationPtge float64, ForeignersPtge float64, SameComAutonPtge float64, SameComAutonDiffProvPtge float64, DifComAutonPtge float64, UnemployLess25_Ptge float64, Unemploy25_40_Ptge float64, UnemployMore40_Ptge float64, AgricultureUnemploymentPtge float64, IndustryUnemploymentPtge float64, ConstructionUnemploymentPtge float64, ServicesUnemploymentPtge float64, totalEmpresas float64, Industria float64, Construcccion float64, ComercTTEHosteleria float64, Servicios float64, ActividadPpal object, inmuebles float64, Pob2010 float64, SUPERFICIE float64, Densidad object, PobChange_pct float64, PersonasInmueble float64, Explotaciones int64

3

Análisis descriptivo del conjunto de datos

El conjunto de datos contiene información demográfica sobre municipios de España junto con los resultados de las últimas elecciones. En concreto, contamos con 8,117 observaciones. Y aunque el archivo no especifica directamente si incluye todos los municipios del país, el tamaño y las variables disponibles sugieren que se cubre casi la totalidad.

Cada observación recoge información en 36 columnas diferentes, vamos a ver cuáles son:

Datos generales

- **Name:** Nombre del municipio.
- **CodigoProvincia:** Código numérico que identifica la provincia a la que pertenece.
- **CCAA:** Comunidad Autónoma a la que pertenece el municipio.

Población y censo

- **Population:** Población total del municipio.
- **TotalCensus:** Número total de personas censadas en el municipio.

Datos electorales

- **AbstentionPtge:** Porcentaje de abstención en las elecciones.
- **AbstencionAlta:** Variable binaria, 1 si la abstención supera el 30%, 0 en caso contrario.

Datos demográficos

- **Porcentajes de población por edades**
 - **Age_0-4_Ptge, Age_under19_Ptge, Age_19_65_pct, Age_over65_pct**
- **WomanPopulationPtge:** Porcentaje de población femenina.
- **ForeignersPtge:** Cambio porcentual de población extranjera.

Datos de movilidad

- **SameComAutonPtge:** Porcentaje de población nacida en la misma comunidad autónoma.
- **SameComAutonDiffProvPtge:** Porcentaje de población nacida en la misma comunidad autónoma, pero en otra provincia.
- **DifComAutonPtge:** Porcentaje de población nacida en una comunidad autónoma diferente.

Datos de desempleo

- **Porcentajes de desempleo por edad**
 - **UnemployLess25_Ptge**, **Unemploy25_40_Ptge**, **UnemployMore40_Ptge**
- **Porcentajes de desempleo por sector**
 - **AgricultureUnemploymentPtge**: Desempleo en el sector agrícola.
 - **IndustryUnemploymentPtge**: Desempleo en el sector industrial.
 - **ConstructionUnemploymentPtge**: Desempleo en el sector de la construcción.
 - **ServicesUnemploymentPtge**: Desempleo en el sector servicios.

Datos económicos

- **totalEmpresas**: Total de empresas registradas.
- **Distribución de empresas por sectores**
 - **Industria**, **Construccion**, **ComercTTEHosteleria**, **Servicios**: Empresas en sectores específicos.

Infraestructura y propiedades

- **inmuebles**: Número de inmuebles registrados.
- **PersonasInmueble**: Promedio de personas por inmueble.

Superficie y densidad

- **SUPERFICIE**: Superficie en kilómetros cuadrados.
- **Densidad**: Clasificación de la densidad (Muy Baja, Baja, Media, etc.).

Otros datos

- **Pob2010**: Población en 2010.
- **PobChange_pct**: Cambio porcentual en la población desde 2010.
- **Explotaciones**: Número de explotaciones registradas en el municipio.

```

1 # Descriptivos de las variables numéricas
2 descriptivos_num = datos.describe().T
3 # Añado más descriptivos a los anteriores
4 for num in numericas:
5     descriptivos_num.loc[num, "Asimetria"] = datos[num].skew()
6     descriptivos_num.loc[num, "Kurtosis"] = datos[num].kurtosis()
7     descriptivos_num.loc[num, "Rango"] = np.ptp(datos[num].dropna().values)

```

Índice	count	mean	std	min	25%	50%	75%	max	Asimetria	Kurtosis	Rango
Population	8117	5722.34	46204.2	5	166	548	2427	3.14199e+06	46.0407	2820.33	3.14199e+06
TotalCensus	8117	4247.86	34423.4	5	140	447	1843	2.36383e+06	46.5446	2893.45	2.36382e+06
AbstentionPtge	8117	26.5016	7.53344	0	21.678	26.424	31.471	57.576	-0.0537359	0.493671	57.576
Izda_Pct	8117	34.404	16.4843	0	21.891	35.165	46.032	94.117	0.0598817	-0.493538	94.117

4

Corrección de los errores detectados

Para comenzar a depurar el conjunto de datos vamos averiguar si existen posibles valores perdidos no identificados, y a asignarlos como tal.

```
1 # Muestro valores perdidos
2 datos[variables].isna().sum()
3
4 # A veces los 'nan' vienen como como una cadena de caracteres, los modifico a perdidos
5 datos[categoricas] = datos[categoricas].replace('nan', np.nan)
6
7 # Missings no declarados de variables cualitativas (NSNC, ?)
8 datos['Densidad'] = datos['Densidad'].replace('?', np.nan)
9
10 # Missings no declarados variables cuantitativas (-1, 99999)
11 datos['Explotaciones'] = datos['Explotaciones'].replace(99999, np.nan)
```

También vamos a recategorizar como perdidos todos los valores fuera de rango, en este caso de los múltiples porcentajes.

```
1 datos['AbstentionPtge'] = [x if 0<=x<=100 else np.nan for x in datos['AbstentionPtge']]
2 datos['Age_0-4_Ptge'] = [x if 0<=x<=100 else np.nan for x in datos['Age_0-4_Ptge']]
3 datos['Age_under19_Ptge'] = [x if 0<=x<=100 else np.nan for x in datos['Age_under19_Ptge']]
4 datos['Age_19_65_pct'] = [x if 0<=x<=100 else np.nan for x in datos['Age_19_65_pct']]
5 datos['Age_over65_pct'] = [x if 0<=x<=100 else np.nan for x in datos['Age_over65_pct']]
6 datos['WomanPopulationPtge'] = [x if 0<=x<=100 else np.nan for x in datos['
  WomanPopulationPtge']]
7 datos['ForeignersPtge'] = [x if -100<=x<=100 else np.nan for x in datos['ForeignersPtge']]
8 datos['SameComAutonPtge'] = [x if 0<=x<=100 else np.nan for x in datos['SameComAutonPtge']]
9 datos['SameComAutonDiffProvPtge'] = [x if 0<=x<=100 else np.nan for x in datos['
  SameComAutonDiffProvPtge']]
10 datos['DifComAutonPtge'] = [x if 0<=x<=100 else np.nan for x in datos['DifComAutonPtge']]
11 datos['UnemployLess25_Ptge'] = [x if 0<=x<=100 else np.nan for x in datos['
  UnemployLess25_Ptge']]
12 datos['Unemploy25_40_Ptge'] = [x if 0<=x<=100 else np.nan for x in datos['Unemploy25_40_Ptge'
  ]]
13 datos['UnemployMore40_Ptge'] = [x if 0<=x<=100 else np.nan for x in datos['
  UnemployMore40_Ptge']]
14 datos['AgricultureUnemploymentPtge'] = [x if 0<=x<=100 else np.nan for x in datos['
  AgricultureUnemploymentPtge']]
15 datos['IndustryUnemploymentPtge'] = [x if 0<=x<=100 else np.nan for x in datos['
  IndustryUnemploymentPtge']]
16 datos['ConstructionUnemploymentPtge'] = [x if 0<=x<=100 else np.nan for x in datos['
  ConstructionUnemploymentPtge']]
17 datos['ServicesUnemploymentPtge'] = [x if 0<=x<=100 else np.nan for x in datos['
  ServicesUnemploymentPtge']]
18 datos['PobChange_pct'] = [x if -100<=x<=100 else np.nan for x in datos['PobChange_pct']]
```

Ahora, vamos a juntar categorías poco representadas de las variables categóricas, en concreto de 'ActividadPpal'. En este caso, agrupar "Industria" y "Construcción" en una sola categoría tiene sentido debido a su baja representación relativa (menos del 0.2% del total), lo que podría generar problemas de significancia estadística y sesgos en el análisis. Además, ambas pertenecen al sector secundario, compartiendo similitudes conceptuales relacionadas con la transformación de materiales e infraestructuras. Este agrupamiento también simplifica el análisis, facilita la interpretación de resultados y previene problemas derivados del desequilibrio en las categorías.

```
1 datos['ActividadPpal'] = datos['ActividadPpal'].replace({'Construccion': 'Const-Ind', '
   Industria': 'Const-Ind'})
2
3 # Indico la variableObj, el ID y las Input
4 datos = datos.set_index(datos['Name']).drop('Name', axis = 1)
5 varObjCont = datos['AbstentionPtge']
6 varObjBin = datos['AbstencionAlta']
7 datos_input = datos.drop(['AbstentionPtge', 'AbstencionAlta'], axis = 1)
8
9 # Genero una lista con los nombres de las variables del conjunto de datos input.
10 variables_input = list(datos_input.columns)
11
12 # Seleciono las variables numéricas
13 numericas_input = list(datos_input.select_dtypes(include = ['int', 'int32', 'int64', 'float',
   'float32', 'float64']).columns)
14
15 # Seleciono las variables categóricas
16 categoricas_input = [variable for variable in variables_input if variable not in
   numericas_input]
```


5

Análisis de valores atípicos (Decisiones)

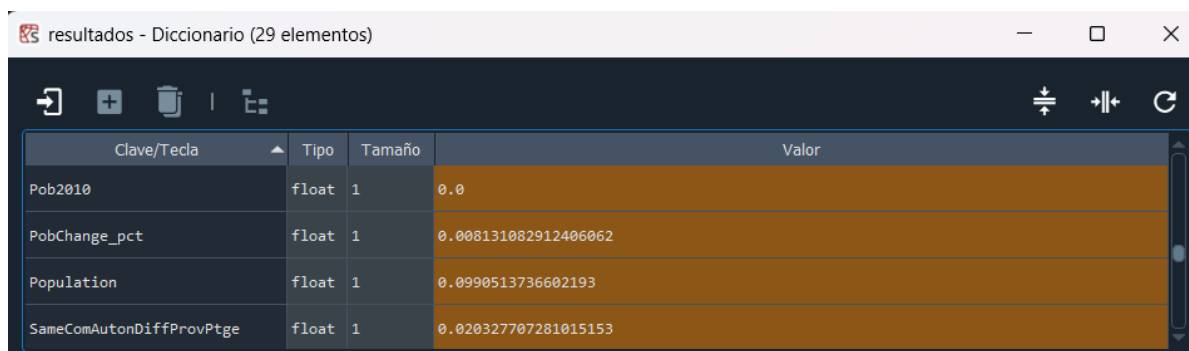
Analicé el porcentaje de valores atípicos en las variables numéricas para identificar posibles anomalías. Los resultados muestran que la mayoría de las variables tienen un porcentaje insignificante de atípicos, lo que indica una alta calidad en los datos.

Sin embargo, destacan Population (9.91%) y TotalCensus (9.63%), probablemente debido a municipios con poblaciones extremadamente pequeñas o grandes, reflejando la heterogeneidad de los datos.

También observé porcentajes elevados en SameComAutonDiffProvPtge (2.03%) y AgricultureUnemploymentPtge (1.99%), posiblemente ligados a características específicas de ciertas regiones. En contraste, variables como ForeignersPtge (0.53%) y ConstructionUnemploymentPtge (0.65%) presentan pocos atípicos, mientras que otras, como Age_0-4_Ptge, ServiciosUnemploymentPtge y PersonasInmueble, no contienen valores atípicos, simplificando su análisis.

Para mitigar el impacto de los valores extremos, transformé los atípicos en valores perdidos. Este tratamiento asegura un análisis robusto y evita distorsiones en los resultados del modelo, preservando la consistencia del conjunto de datos.

```
1 # Cuento el porcentaje de atipicos de cada variable
2 resultados = {x: atipicosAmissing(datos_input[x])[1] / len(datos_input) for x in
    numericas_input}
3
4 # Modifico los atipicos como missings
5 for x in numericas_input:
6     datos_input[x] = atipicosAmissing(datos_input[x])[0]
```

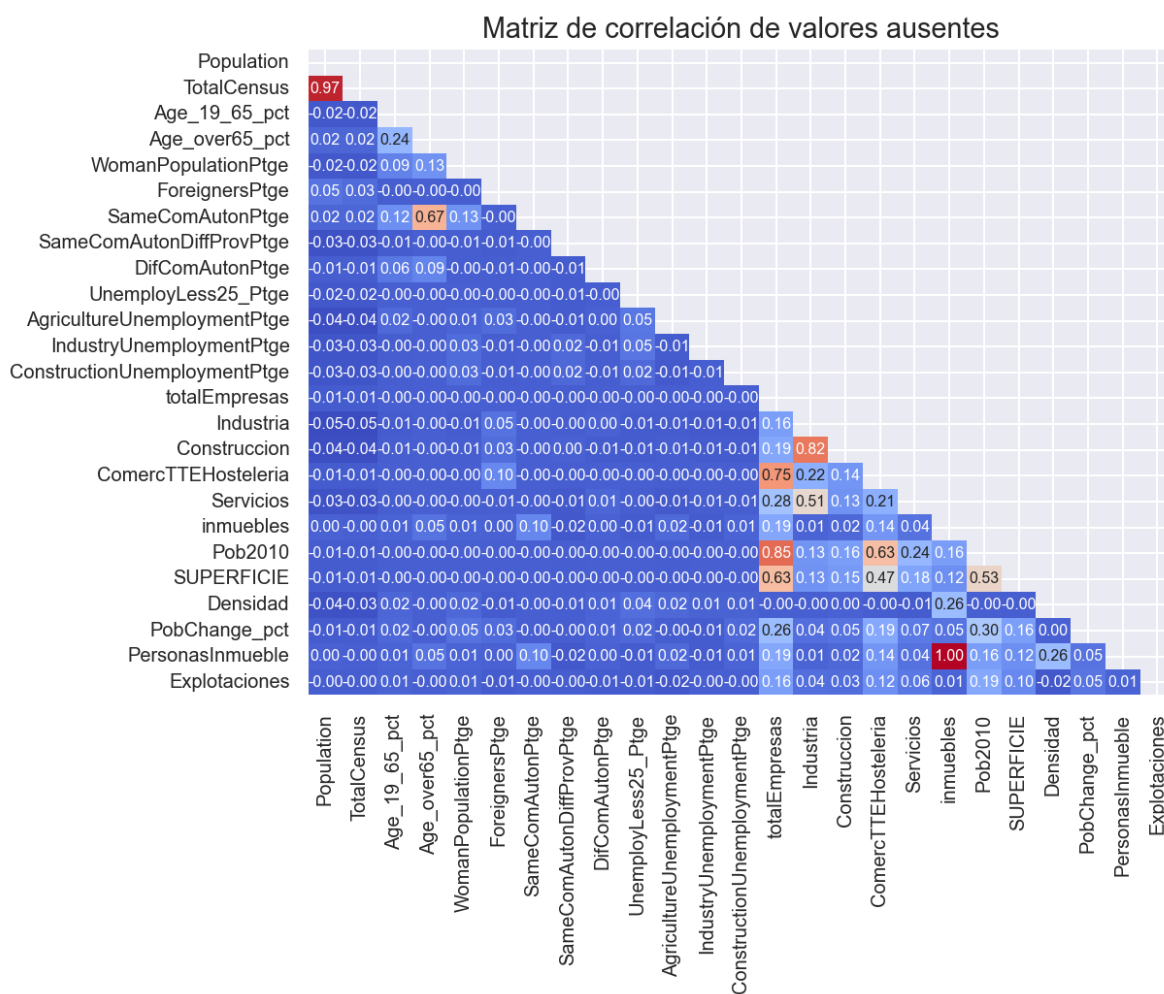


Clave/Tecla	Tipo	Tamaño	Valor
Pob2010	float	1	0.0
PobChange_pct	float	1	0.008131082912406062
Population	float	1	0.0990513736602193
SameComAutonDiffProvPtge	float	1	0.020327707281015153

6

Análisis de valores perdidos (Imputaciones)

```
1 # Visualizo un mapa de calor con la matriz de correlación de valores ausentes en los datos
2 patron_perdidos(datos_input)
```



```
1 # Muestro el total de valores perdidos por cada variable
2 datos_input[variables_input].isna().sum()
```

Out[]: Population 804, TotalCensus 782, Age_19_65_pct 24, Age_over65_pct 3, WomanPopulationPtge 21, ForeignersPtge 43, SameComAutonPtge 3, SameComAutonDiffProvPtge 165, DifComAutonPtge 40, UnemployLess25_Ptge 26, AgricultureUnemploymentPtge 162, IndustryUnemploymentPtge 48, ConstructionUnemploymentPtge 53, totalEmpresas 5, Industria 188, Construcccion 139, ComercTTE-Hosteleria 9, Servicios 62, inmuebles 138, Pob2010 7, SUPERFICIE 8, Densidad 92, PobChange_pct 75, PersonasInmueble 138, Explotaciones 189

```
1 # Guardo la proporción de valores perdidos por variable en una nueva variable
2 prop_missingsVars = datos_input.isna().sum()/len(datos_input)
3
4 # Creo la variable prop_missings que recoge el número de valores perdidos por cada
  observación
5 datos_input['prop_missings'] = datos_input.isna().mean(axis = 1)
6
7 # Realizo un estudio descriptivo básico a la nueva variable
8 datos_input['prop_missings'].describe()
```

Out[]: count 8117.000000, mean 0.012036, std 0.024653, min 0.000000, 25% 0.000000, 50% 0.000000, 75% 0.000000, max 0.333333

```
1 # Calculo el número de valores distintos que tiene la nueva variable (tiene 9)
2 # No la transformo en categórica pues al hacer la partición de los datos, no hay
  representación de todas las categorías en los conjuntos de train y test entonces al crear
  las dummies se generan diferente número de columnas y no se puede calcular la pseudo R^2
3 len(datos_input['prop_missings'].unique())
4
5 # Elimino las observaciones con mas de la mitad de datos missings
6 eliminar = datos_input['prop_missings'] > 0.5
7 datos_input = datos_input[~eliminar]
8 varObjBin = varObjBin[~eliminar]
9 varObjCont = varObjCont[~eliminar]
10
11 # Agrego 'prop_missings' a la lista de nombres de variables input
12 variables_input.append('prop_missings')
13 numericas_input.append('prop_missings')
14
15 # Elimino las variables con mas de la mitad de datos missings (no hay ninguna)
16 eliminar = [prop_missingsVars.index[x] for x in range(len(prop_missingsVars)) if
  prop_missingsVars[x] > 0.5]
17 datos_input = datos_input.drop(eliminar, axis = 1)
```

No se considera necesario recategorizar las variables categóricas analizadas porque todas presentan una distribución suficientemente representativa o ya están estructuradas de manera adecuada para el análisis. Por ejemplo, variables como CódigoProvincia y CCAA ofrecen información detallada y diferenciada por regiones geográficas, cuya agregación podría llevar a la pérdida de información relevante. Además, la variable Densidad, aunque categórica, tiene tres niveles bien distribuidos y suficientemente representativos, lo que no compromete la robustez de los análisis posteriores.

6.1. Imputaciones

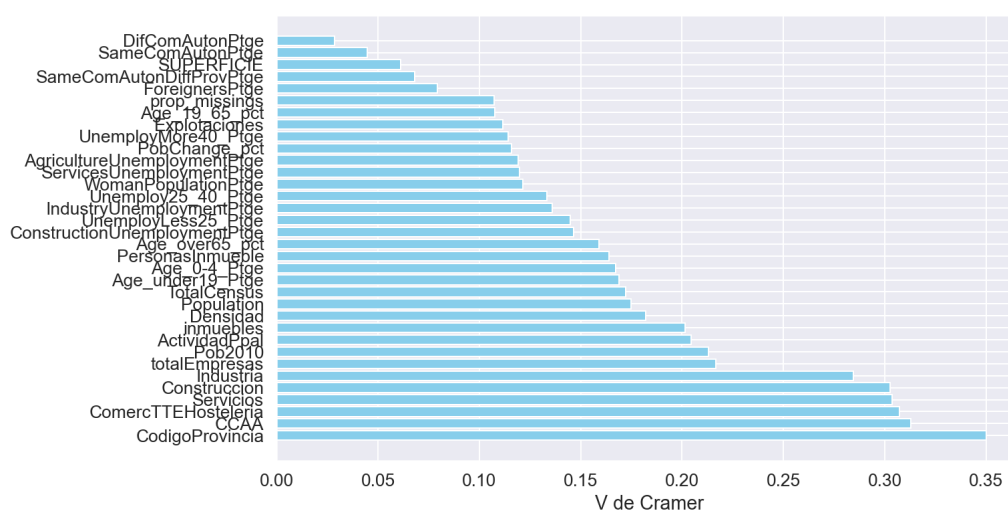
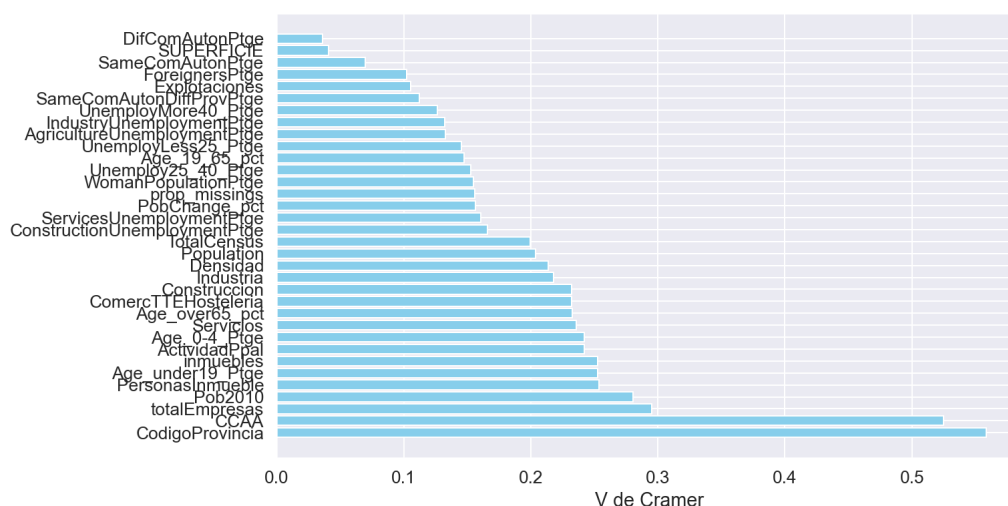
En esta sección, imputo los valores perdidos de las variables cuantitativas y cualitativas utilizando el método aleatorio para preservar las distribuciones originales. Tras el proceso, todas las columnas quedan sin valores faltantes.

```
1 # Imputo todas las cuantitativas, seleccionando el tipo de imputacion: aleatorio
2 for x in numericas_input:
3     datos_input[x] = ImputacionCuant(datos_input[x], 'aleatorio')
4
5 # Imputo todas las cualitativas, seleccionando el tipo de imputacion: aleatorio
6 for x in categoricas_input:
7     datos_input[x] = ImputacionCuali(datos_input[x], 'aleatorio')
8 # Reviso que no queden datos missings
9 datos_input.isna().sum()
```

7

Detección de las relaciones entre las variables

```
1 # Obtengo la importancia de las variables
2 graficoVcramer(datos_input, varObjBin)
3 graficoVcramer(datos_input, varObjCont)
```



```

1 # Crear un DataFrame para almacenar los resultados del coeficiente V de Cramer
2 VCramer = pd.DataFrame(columns=['Variable', 'Objetivo', 'Vcramer'])
3
4 for variable in variables_input:
5     v_cramer = VCramer(datos_input[variable], varObjCont)
6     VCramer = VCramer.append({'Variable': variable, 'Objetivo': varObjCont.name, 'Vcramer':
7         v_cramer}, ignore_index=True)
8
9 for variable in variables_input:
10    v_cramer = VCramer(datos_input[variable], varObjBin)
11    VCramer = VCramer.append({'Variable': variable, 'Objetivo': varObjBin.name, 'Vcramer':
12        v_cramer}, ignore_index=True)

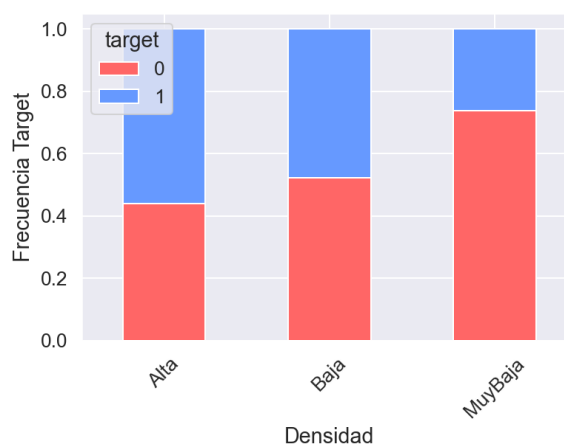
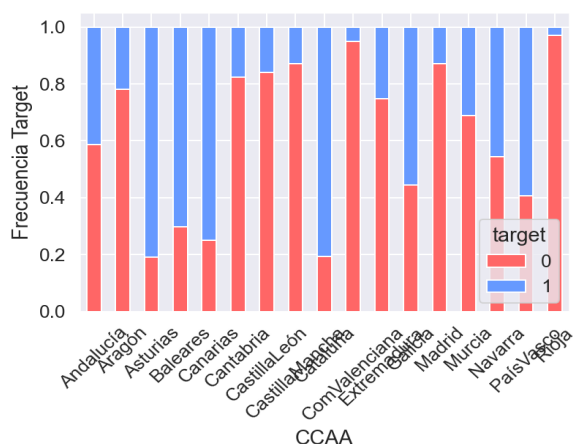
```

7.1. Detección de las relaciones entre todas las variables input y la variable objetivo binaria

```

1 # Veo graficamente el efecto de dos variables cualitativas sobre la binaria
2 # Tomo las variables con más y menos relación con la variable objetivo Binaria
3 mosaico_targetbinaria(datos_input['CodigoProvincia'], varObjBin, 'CodigoProvincia')
4 mosaico_targetbinaria(datos_input['Izquierda'], varObjBin, 'Izquierda')

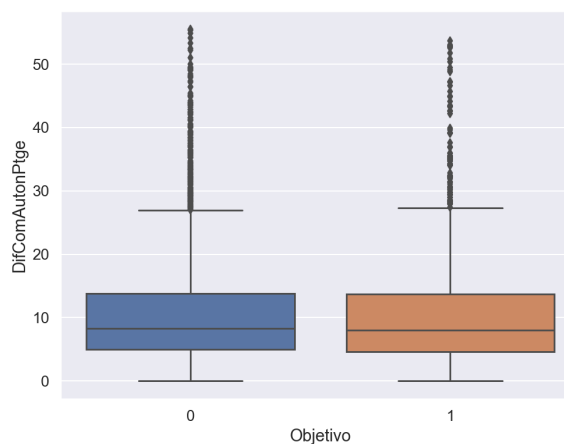
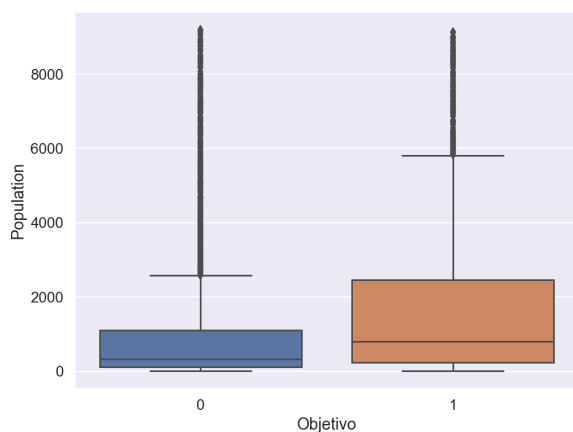
```



```

1 # Veo graficamente el efecto de dos variables cuantitativas sobre la binaria
2 boxplot_targetbinaria(datos_input['Population'], varObjBin, 'Objetivo', 'Population')
3 boxplot_targetbinaria(datos_input['DifComAutonPtge'], varObjBin, 'Objetivo', 'DifComAutonPtge')

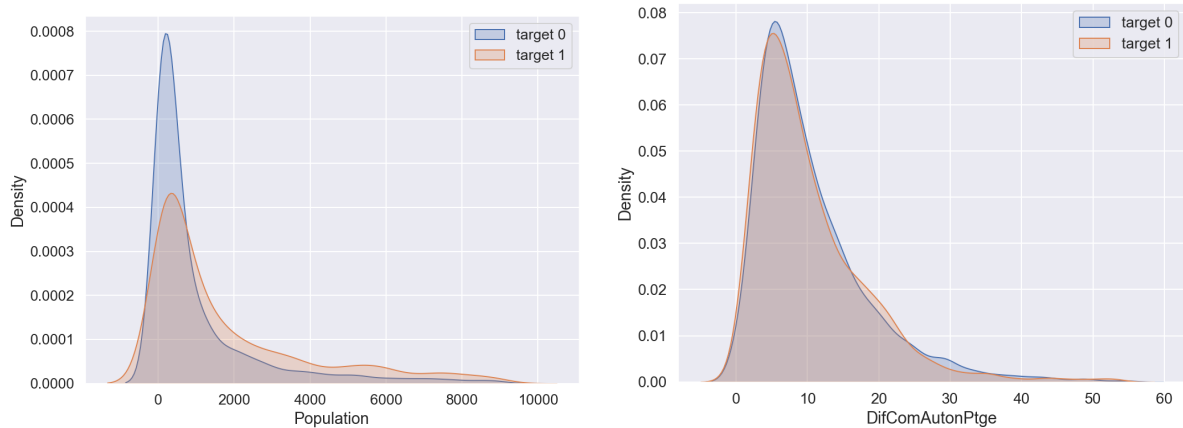
```



```

1 hist_targetbinaria(datos_input['Population'], varObjBin, 'Population')
2 hist_targetbinaria(datos_input['DifComAutonPtge'], varObjBin, 'DifComAutonPtge')

```

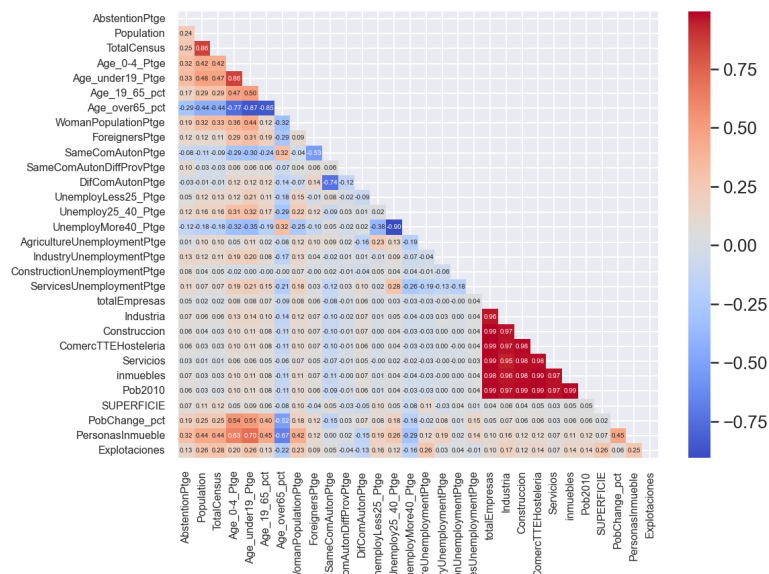


7.2. Detección de las relaciones entre todas las variables input y la variable objetivo continua

```

1 # Correlación entre todas las variables numéricas frente a la objetivo continua
2 numericas = datos_input.select_dtypes(include=['int', 'float']).columns
3 matriz_corr = pd.concat([varObjCont, datos_input[numericas]], axis = 1).corr(method = '
    pearson')
4 mask = np.triu(np.ones_like(matriz_corr, dtype=bool))
5 plt.figure(figsize=(8, 6))
6 sns.set(font_scale=1.2)
7 sns.heatmap(matriz_corr, annot=True, annot_kws={"size": 5}, cmap='coolwarm', fmt=".2f", cbar=
    True, mask=mask)
8 plt.yticks(ticks=np.arange(len(matriz_corr.index)) + 0.5, labels=matriz_corr.index, rotation
    =0, fontsize=8)
9 plt.xticks(ticks=np.arange(len(matriz_corr.columns)) + 0.5, labels=matriz_corr.columns,
    rotation=90, fontsize=8)
10 plt.show()

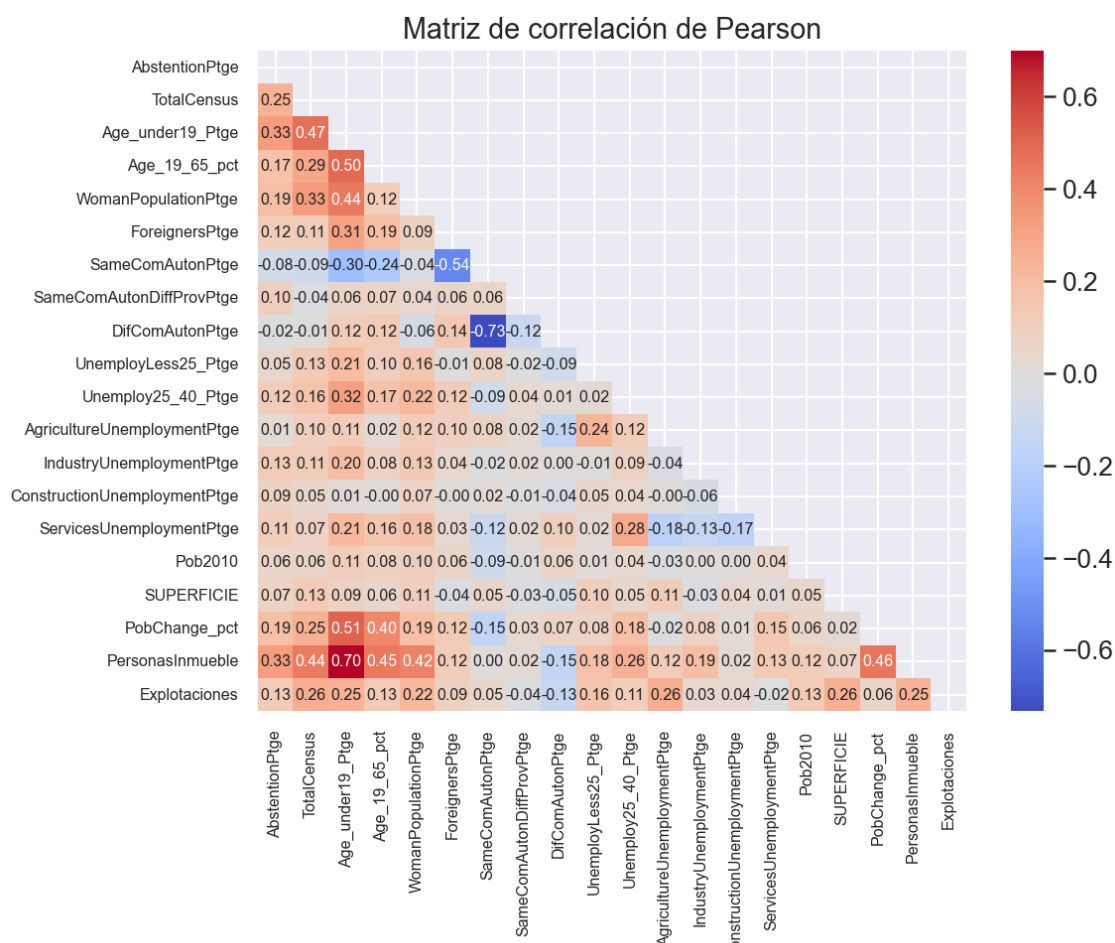
```



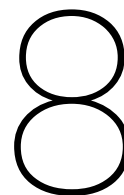
En el análisis realizado a través del gráfico de la matriz de correlación, se identificaron claros indicios de multicolinealidad. Para abordar este problema, decidí eliminar aquellas variables redundantes que presentaban una alta correlación entre sí. En particular, las variables demográficas relacionadas con la distribución etaria, como Age_0-4_Ptge y Age_over65_pct, así como las variables vinculadas con porcentajes de desempleo por edad, como UnemployMore40_Ptge, mostraron una dependencia lineal perfecta. Esto es habitual cuando las variables representan proporciones cuya suma es igual al 100%, como ocurre con los grupos de edad o las categorías de desempleo, ya que una variable puede ser fácilmente derivada de las otras, lo que las convierte en redundantes. De manera similar, variables como totalCensus y Population presentaban una correlación extremadamente alta, superior a 0.8 en el coeficiente de Pearson, lo que indicaba que ambas variables aportaban información similar y, por lo tanto, una de ellas resultaba innecesaria. También eliminé algunas variables sectoriales, tales como Industria, Construcción, Servicios, ComercTEHosteleria, totalEmpresas, y otras variables como inmuebles. Con estas eliminaciones, optimicé el modelo para reducir la multicolinealidad y mantener solo aquellas variables que capturan de manera más efectiva la variabilidad de los datos.

```
1 datos_input = datos_input.drop(['Industria', 'Construccion', 'Servicios', 'Population', '
    totalEmpresas', 'inmuebles', 'ComercTEHosteleria', 'Age_0-4_Ptge', 'UnemployMore40_Ptge'
    , 'Age_over65_pct'], axis=1)
```

Este proceso asegura un modelo más parsimonioso y estable, con menor redundancia y mejores propiedades estadísticas. La matriz de correlación de nuestro nuevo modelo queda así:



```
1 # Una vez finalizado este proceso, considero que los datos estan depurados. Los guardo
2 datosEleccionesDep = pd.concat([varObjBin, varObjCont, datos_input], axis = 1)
3 with open('datosEleccionesDep.pickle', 'wb') as archivo:
4     pickle.dump(datosEleccionesDep, archivo)
```



Construcción del modelo de regresión lineal

Para explicar y predecir la variabilidad de una variable continua, en este caso el porcentaje de abstención electoral (AbstentionPtge), construiré un modelo de regresión lineal seleccionando cuidadosamente las variables predictoras más relevantes. Mi objetivo es garantizar un equilibrio entre simplicidad y precisión en el modelo. Para ello, utilizaré métodos clásicos de selección, los cuales proporcionan un enfoque sistemático para identificar el subconjunto óptimo de variables. Entre los métodos considerados destacan la selección hacia adelante, que comienza con un modelo vacío y añade variables de forma incremental; la selección hacia atrás, que inicia con un modelo completo y elimina variables de manera gradual; y la selección por pasos, que combina ambos enfoques. En todos los casos, me basaré en criterios estadísticos como el AIC o el BIC para decidir qué variables incluir o excluir. Este proceso garantiza la construcción de un modelo eficiente, maximizando el ajuste a los datos sin introducir variables innecesarias que puedan complicar la interpretación.

En este análisis, no aplicaré transformaciones a las variables predictoras, de modo que la interpretación de los resultados sea directa y sencilla. Además, solo consideraré interacciones entre las variables continuas para explorar posibles efectos combinados, manteniendo así un enfoque claro y focalizado en los factores que contribuyen a explicar la variabilidad en la abstención electoral.

```
1 # Identifico las variables continuas y categóricas
2 var_cont = ['TotalCensus', 'Age_under19_Ptge', 'Age_19_65_pct', 'WomanPopulationPtge', '
  ForeignersPtge', 'SameComAutonPtge', 'SameComAutonDiffProvPtge', 'DifComAutonPtge', '
  UnemployLess25_Ptge', 'Unemploy25_40_Ptge', 'AgricultureUnemploymentPtge', '
  IndustryUnemploymentPtge', 'ConstructionUnemploymentPtge', 'ServicesUnemploymentPtge', '
  Pob2010', 'SUPERFICIE', 'PobChange_pct', 'PersonasInmuable', 'Explotaciones']
3 var_categ = ['CodigoProvincia', 'CCAA', 'ActividadPpal', 'Densidad', 'prop_missings']
4
5 # Hago la particion
6 x_train, x_test, y_train, y_test = train_test_split(datos_input, varObjCont, test_size = 0.2,
  random_state = 1234567)
7 [language=python]
```

He seleccionado las mejores variables continuas basándome en su V de Cramer, priorizando aquellas que tienen la mayor asociación con la variable dependiente. Dado que la capacidad computacional de mi ordenador es limitada, no es viable crear interacciones entre todas las variables continuas disponibles, ya que esto implicaría calcular combinaciones exponencialmente más complejas. Por lo tanto, elegir un subconjunto de las mejores variables permite reducir la complejidad del modelo y optimizar los recursos sin sacrificar demasiada precisión en los resultados. Estas variables representan las que tienen mayor probabilidad de aportar información relevante al modelo, maximizando la eficiencia del análisis.


```

1 # Interacciones 2 a 2 sólo de las variables continuas seleccionadas
2 interacciones = ['Pob2010', 'TotalCensus', 'Age_under19_Ptge', 'PersonasInmueble', '
    ConstructionUnemploymentPtge', 'UnemployLess25_Ptge', 'IndustryUnemploymentPtge', '
    Unemploy25_40_Ptge']
3 interacciones_unicas = list(itertools.combinations(interacciones, 2))
4
5 modeloBackAIC=lm_backward(y_train, x_train, var_cont, var_categ, interacciones_unicas, 'AIC')
6 modeloBackBIC=lm_backward(y_train, x_train, var_cont, var_categ, interacciones_unicas, 'BIC')
7 modeloForwAIC=lm_forward(y_train, x_train, var_cont, var_categ, interacciones_unicas, 'AIC')
8 modeloForwBIC=lm_forward(y_train, x_train, var_cont, var_categ, interacciones_unicas, 'BIC')
9 modeloStepAIC=lm_stepwise(y_train, x_train, var_cont, var_categ, interacciones_unicas, 'AIC')
10 modeloStepBIC=lm_stepwise(y_train, x_train, var_cont, var_categ, interacciones_unicas, 'BIC')

```

Método	Métrica	R ² Train	R ² Test	Nº Parámetros
Backward	AIC	0.4331611196273921	0.42085246054061676	74
Backward	BIC	0.4284799832657915	0.4152402475009853	66
Forward	AIC	0.43102798042448465	0.41715290012509343	75
Forward	BIC	0.42439754619792025	0.41140944572866434	62
Stepwise	AIC	0.43063103243507705	0.4191446754593715	72
Stepwise	BIC	0.42439754619792025	0.41140944572866434	62

Table 8.1: Comparación de métodos de selección clásica de variables, con interacciones.

De acuerdo con los resultados obtenidos, el método seleccionado sería el método Forward con BIC, ya que logra un equilibrio óptimo entre rendimiento y simplicidad. Este modelo presenta un R^2 de test de 0.4114, que se encuentra entre los valores más altos observados, utilizando únicamente 62 parámetros. El uso del criterio BIC resulta particularmente adecuado, ya que penaliza de manera más estricta la complejidad del modelo en comparación con el AIC, lo que favorece la selección de un modelo más parsimonioso y con mayor capacidad de generalización en datos nuevos. Además, aunque el método Stepwise con BIC tiene el mismo rendimiento y número de parámetros, Forward con BIC es preferible porque sigue un enfoque incremental que garantiza la inclusión progresiva de variables relevantes, reduciendo el riesgo de seleccionar aquellas que sean menos significativas. Este enfoque no solo permite construir un modelo eficiente, sino también asegura interpretaciones claras, lo que resulta ideal para un análisis como este, centrado en explorar y entender los factores asociados a los niveles de abstención electoral.

Consecutivamente, implemento la **selección de variables aleatoria** como una estrategia para generar submuestras aleatorias a partir de los datos originales con el método Forward con BIC.. Este enfoque me permite identificar patrones recurrentes en las variables seleccionadas y determinar cuáles son las más relevantes y estables en distintos escenarios de entrenamiento. Al priorizar los modelos más frecuentes en estas submuestras, puedo evaluar de manera detallada su consistencia y rendimiento, asegurando que el modelo final sea robusto y generalizable a nuevos datos.

```

1 # Inicializo un diccionario para almacenar las fórmulas y variables seleccionadas.
2 variables_seleccionadas = {
3     'Formula': [],
4     'Variables': []}
5
6 # Realizo 30 iteraciones de selección aleatoria.
7 for x in range(30):
8     print('-----iter: ' + str(x))
9     # Divido los datos de entrenamiento en conjuntos de entrenamiento y prueba.
10    x_train2, x_test2, y_train2, y_test2 = train_test_split(x_train, y_train, test_size =
        0.3, random_state = 1234567 + x)
11    # Realizo la selección forward utilizando el criterio BIC en la submuestra.
12    modelo = lm_forward(y_train2.astype(int), x_train2, var_cont, var_categ,
        interacciones_unicas, 'BIC')
13    # Almaceno las variables seleccionadas y la fórmula correspondiente.
14    variables_seleccionadas['Variables'].append(modelo['Variables'])
15    variables_seleccionadas['Formula'].append(sorted(modelo['Modelo'].model.exog_names))
16 # Uno las variables en las fórmulas seleccionadas en una sola cadena.
17 variables_seleccionadas['Formula'] = list(map(lambda x: '+'.join(x), variables_seleccionadas[
    'Formula']))

```

Una vez finalizadas las iteraciones, calculo las frecuencias de las fórmulas seleccionadas en las 30 repeticiones. Las tres fórmulas más frecuentes son identificadas como candidatas. Las variables correspondientes a estas fórmulas se extraen para su análisis posterior. Este enfoque asegura que los modelos seleccionados representen configuraciones recurrentes y, por ende, sean más estables.

```

1 # Calculo la frecuencia de cada fórmula y las ordeno por frecuencia.
2 frecuencias = Counter(variables_seleccionadas['Formula'])
3 frec_ordenada = pd.DataFrame(list(frecuencias.items()), columns = ['Formula', 'Frecuencia'])
4 frec_ordenada = frec_ordenada.sort_values('Frecuencia', ascending = False).reset_index()
5
6 # Identifico los tres modelos más frecuentes y las variables correspondientes.
7 var_1 = variables_seleccionadas['Variables'][variables_seleccionadas['Formula'].index(
8     frec_ordenada['Formula'][0])]
9 var_2 = variables_seleccionadas['Variables'][variables_seleccionadas['Formula'].index(
10     frec_ordenada['Formula'][1])]
11 var_3 = variables_seleccionadas['Variables'][variables_seleccionadas['Formula'].index(
12     frec_ordenada['Formula'][2])]

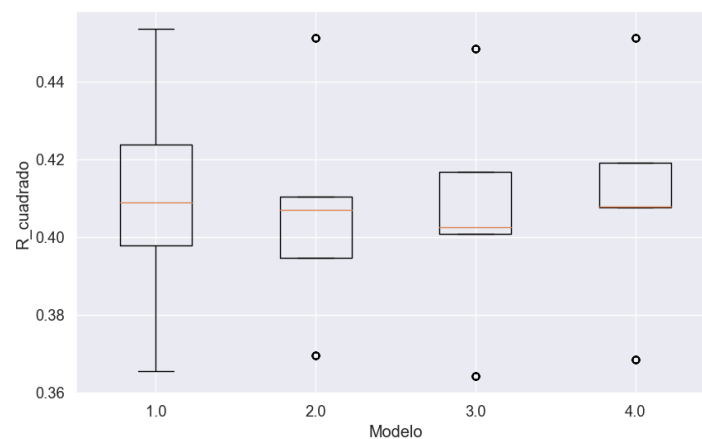
```

Con los modelos candidatos (el modelo ganador inicial y las tres alternativas más frecuentes), realizo una **validación cruzada** de 5 pliegues en 20 repeticiones. Este procedimiento evalúa el desempeño de cada modelo en términos de su R^2 medio, proporcionando una estimación robusta de su capacidad predictiva. Los resultados se visualizan a través de un boxplot que permite comparar gráficamente la distribución de R^2 para cada modelo.

```

1 results = pd.DataFrame({'Rsquared': [], 'Resample': [], 'Modelo': []})
2 for rep in range(20):
3     modelo1=validacion_cruzada_lm(5,x_train,y_train,modeloForwBIC['Variables']['cont'],
4     modeloForwBIC['Variables']['categ'], modeloForwBIC['Variables']['inter'])
5     modelo2=validacion_cruzada_lm(5,x_train,y_train,var_1['cont'],var_1['categ'],var_1['inter'])
6     modelo3=validacion_cruzada_lm(5,x_train,y_train,var_2['cont'],var_2['categ'],var_2['inter'])
7     modelo4=validacion_cruzada_lm(5,x_train,y_train,var_3['cont'],var_3['categ'],var_3['inter'])
8     results_rep = pd.DataFrame({
9         'Rsquared': modelo1 + modelo2 + modelo3 + modelo4,
10        'Resample': ['Rep' + str((rep + 1))*5*4,
11        'Modelo': [1]*5 + [2]*5 + [3]*5 + [4]*5})
12    results = pd.concat([results, results_rep], axis = 0)
13
14 # Boxplot de la validacion cruzada
15 plt.figure(figsize=(10, 6))
16 plt.grid(True)
17 grupo_metrica = results.groupby('Modelo')['Rsquared']
18 boxplot_data = [grupo_metrica.get_group(grupo).tolist() for grupo in grupo_metrica.groups]
19 plt.boxplot(boxplot_data, labels=grupo_metrica.groups.keys())
20 plt.xlabel('Modelo')
21 plt.ylabel('R_cuadrado')
22 plt.show()

```



```

1 # Calcular la media de las métricas R-squared por modelo
2 media_r2 = results.groupby('Modelo')['Rsquared'].mean()
3 print(media_r2)

```

Out[]: 1.0 → 0.409870, 2.0 → 0.406500, 3.0 → 0.406521, 4.0 → 0.410818

```

1 # Calcular la desviación estándar de las métricas R-squared por modelo
2 std_r2 = results.groupby('Modelo')['Rsquared'].std()
3 print(std_r2)

```

Out[]: 1.0 → 0.029191, 2.0 → 0.026672, 3.0 → 0.027381, 4.0 → 0.026663

```

1 # Contar el número de parámetros en cada modelo
2 num_params = [len(modeloForwBIC['Modelo'].params),
3               len(frec_ordenada['Formula'][0].split('+')),
4               len(frec_ordenada['Formula'][1].split('+')),
5               len(frec_ordenada['Formula'][2].split('+'))]
6 print(num_params)

```

Out[]: 1.0 → 62, 2.0 → 57, 3.0 → 58, 4.0 → 62

Basándose estrictamente en la métrica de desempeño (el valor medio del R^2 y observando la dispersión (desviación estándar), el modelo 4 se perfila como el “mejor” entre los cuatro, pues es el que obtiene el R^2 medio más alto (0.410818), superando muy ligeramente al modelo 1 (0.409870) y de forma un poco más amplia a los modelos 2 y 3 (en torno a 0.4065). La desviación estándar de su R^2 (0.026663) no es mayor que la de los otros modelos (de hecho, es prácticamente la más baja junto con la de los modelos 2 y 3), lo cual sugiere que su rendimiento no solo es algo superior en promedio, sino también estable o al menos no más inestable que el resto. Tiene el mismo número de parámetros que el modelo 1 (62), así que, a igual complejidad, ofrece mejor R^2 . Los modelos 2 y 3 son algo más simples (57 y 58 parámetros) pero también con un R^2 medio ligeramente menor.

Por último, evalúo el modelo ganador (4.0) en términos de sus coeficientes, la significancia estadística de sus parámetros y su capacidad predictiva en los datos de entrenamiento y prueba.

```

1 ModeloGanador = lm_forward(y_train, x_train, var_3['cont'], var_3['categ'], var_3['inter'], '
  BIC')
2 # Muestro el resumen de statsmodels
3 ModeloGanador['Modelo'].summary()

```

OLS Regression Results							
Dep. Variable:	AbstentionPtge	R-squared:	0.424				
Model:	OLS	Adj. R-squared:	0.418				
Method:	Least Squares	F-statistic:	78.83				
Date:	Sat, 25 Jan 2025	Prob (F-statistic):	0.00				
Time:	20:16:51	Log-Likelihood:	-20496.				
No. Observations:	6493	AIC:	4.111e+04				
Df Residuals:	6432	BIC:	4.153e+04				
Df Model:	60						
Covariance Type:	nonrobust						
		coef	std err	t	P> t	[0.025	0.975]
	const	39.0833	1.403	27.848	0.000	36.332	41.834
	SameComAutonPtge	-0.0937	0.008	-12.057	0.000	-0.109	-0.078
	Explotaciones	0.0025	0.000	6.711	0.000	0.002	0.003
	ConstructionUnemploymentPtge	0.0307	0.007	4.475	0.000	0.017	0.044

	ActividadPpal_Otro	-1.5748	0.258	-6.100	0.000	-2.081	-1.069
	ActividadPpal_Servicios	-0.8640	0.309	-2.793	0.005	-1.470	-0.258
	TotalCensus_PersonasInmueble	-0.0006	0.000	-5.475	0.000	-0.001	-0.000
	Age_under19_Ptge_PersonasInmueble	0.0329	0.008	3.876	0.000	0.016	0.050
Omnibus:	249.944	Durbin-Watson:	1.977				
Prob(Omnibus):	0.000	Jarque-Bera (JB):	669.088				
Skew:	0.157	Prob(JB):	5.12e-146				
Kurtosis:	4.541	Cond. No.	3.07e+05				

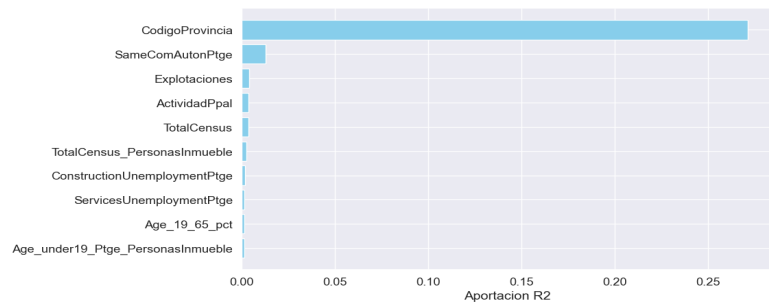
En este modelo, la mayoría de los parámetros son estadísticamente significativos ($p < 0.05$), lo que indica que variables como SameComAutonPtge, los porcentajes de desempleo en distintos sectores, y varias provincias (CodigoProvincia) tienen un impacto relevante en la abstención electoral. Por ejemplo, SameComAutonPtge reduce la abstención, mientras que sectores como ConstructionUnemploymentPtge y IndustryUnemploymentPtge la incrementan. Sin embargo, algunas variables, como CodigoProvincia_18 y ActividadPpal_Const-Ind, no son significativas y podrían excluirse en futuras iteraciones para mejorar la simplicidad y eficacia del modelo.

Interpretación de coeficientes:

- Variable continua (SameComAutonPtge): Tiene un coeficiente de -0.0937, lo que implica que por cada aumento del 1% en el porcentaje de personas nacidas en la misma comunidad autónoma, la abstención disminuye en un 0.094%, manteniendo constantes las demás variables. Este resultado resalta la importancia del sentido de pertenencia territorial en la participación electoral.
- Variable binaria (ActividadPpal_Servicios): Su coeficiente es -0.864, indicando que los municipios cuya actividad principal es el sector servicios tienen una reducción promedio de 0.86% en la abstención, en comparación con la categoría base. Esto sugiere que economías orientadas a servicios pueden favorecer una mayor participación.

```
1 # Evaluamos la estabilidad del modelo a partir de las diferencias en train y test
2 Rsq(ModeloGanador['Modelo'], y_train, ModeloGanador['X']) #Out[: 0.4237612287216389
3 x_test_modeloganador = crear_data_modelo(x_test, ModeloGanador['Variables']['cont'],
4     ModeloGanador['Variables']['categ'], ModeloGanador['Variables']['inter'])
5 Rsq(ModeloGanador['Modelo'], y_test, x_test_modeloganador) #Out[: 0.41255161298279974
6 modelEffectSizes(ModeloGanador, y_train, x_train, ModeloGanador['Variables']['cont'],
7     ModeloGanador['Variables']['categ'], ModeloGanador['Variables']['inter'])
```

	Variables	R2
9	Age_under19_Ptge_PersonasInmueble	0.001346
5	Age_19_65_pct	0.001361
3	ServicesUnemploymentPtge	0.001444
2	ConstructionUnemploymentPtge	0.001794
8	TotalCensus_PersonasInmueble	0.002686
4	TotalCensus	0.003538
7	ActividadPpal	0.003568
1	Explotaciones	0.004034
0	SameComAutonPtge	0.013024
6	CodigoProvincia	0.271546



El modelo ganador, lo selecciono por su buen balance entre rendimiento y parsimonia. Presenta un R^2 ajustado de 0.413, lo que indica que explica el 41.3% de la variabilidad en la abstención electoral, un valor sólido para un modelo con 62 parámetros. Además, las diferencias entre los R^2 de entrenamiento (0.4237) y prueba (0.4125) son mínimas, lo que evidencia una buena capacidad de generalización y ausencia de sobreajuste.

9

Construcción del modelo de regresión logística.

En esta sección se aborda la construcción del modelo de regresión logística, una herramienta estadística ampliamente utilizada para analizar y predecir la probabilidad de ocurrencia de un evento binario. En este caso, el objetivo del modelo es estimar la probabilidad de que un municipio presente una alta abstención electoral, definida como un porcentaje superior al 30% (AbstenciónAlta).

El desarrollo del modelo incluye varias etapas, algunas de las cuáles resultan simétricas a las del modelo de regresión lineal: desde la preparación y transformación de los datos, la partición de estos mismos en entrenamiento y prueba, la selección de las interacciones, etcétera. Es por eso que en esta primera parte voy a pasarlo todo un poco por encima.

```
1 # Cargo los datos depurados
2 with open('datosEleccionesDep.pickle', 'rb') as f:
3     datos_input = pickle.load(f)
4
5 # Identifico la variable objetivo y la elimino de mi conjunto de datos.
6 varObjBin = datos_input['AbstencionAlta']
7 datos_input = datos_input.drop(['AbstencionPtge', 'AbstencionAlta'], axis = 1)
8
9 # Veo el reparto original. Compruebo que la variable objetivo tome valor 1 para el evento y 0
   para el no evento
10 pd.DataFrame({
11     'n': varObjBin.value_counts()
12     , '%': varObjBin.value_counts(normalize = True)
13 })
14
15 # Genero una lista con los nombres de las variables del conjunto de datos input.
16 variables_input = list(datos_input.columns)
17
18 # Seleciono las variables numéricas y categóricas
19 var_cont = list(datos_input.select_dtypes(include = ['int', 'int32', 'int64', 'float', '
   float32', 'float64']).columns)
20 var_categ = [variable for variable in variables_input if variable not in var_cont]
21
22 # Obtengo la particion
23 x_train, x_test, y_train, y_test = train_test_split(datos_input, varObjBin, test_size = 0.2,
   random_state = 1234567)
24 y_train, y_test = y_train.astype(int), y_test.astype(int)
25
26 # Interacciones 2 a 2 sólo de las variables continuas seleccionadas con criterio de la V de
   Cramer de estas mismas, por temas de capacidad computacional
27 graficoVcramer(datos_input, varObjBin)
28 interacciones = ['Pob2010', 'PersonasInmueble', 'Age_under19_Ptge']
29
30 var_inter = list(itertools.combinations(interacciones, 2))
```

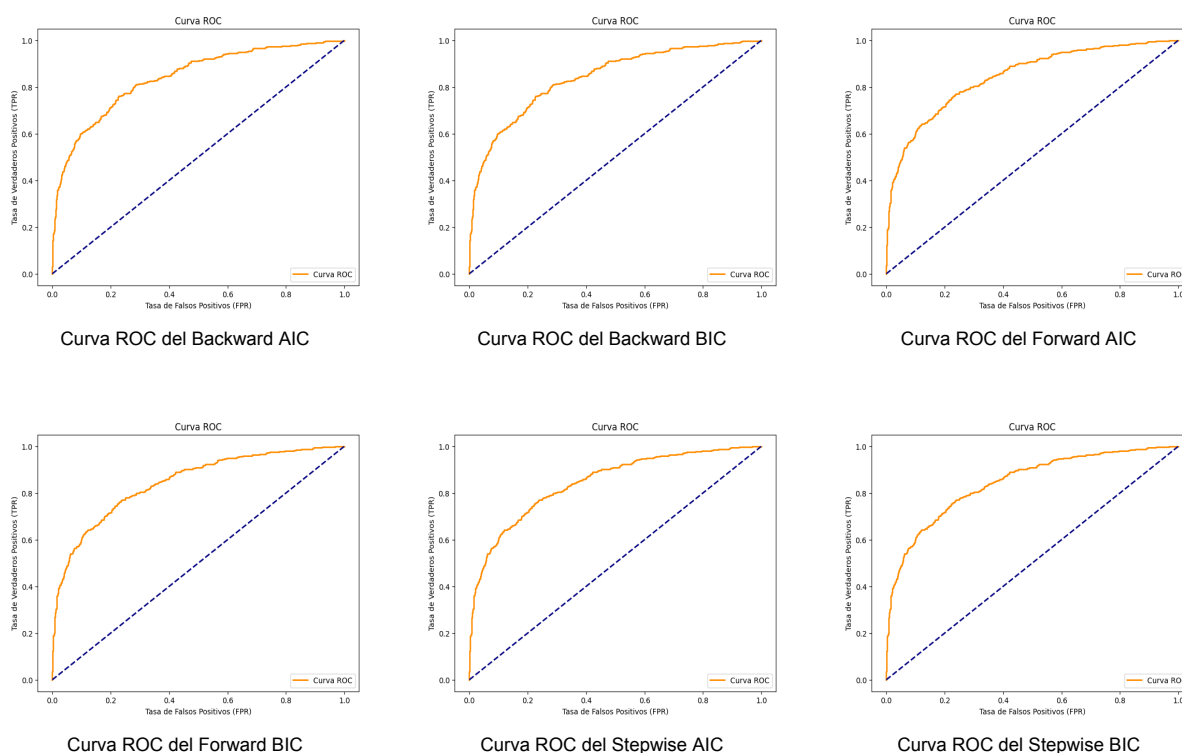
```

1 modeloBackAIC = glm_backward(y_train, x_train, var_cont, var_categ, var_inter, 'AIC')
2 modeloBackBIC = glm_backward(y_train, x_train, var_cont, var_categ, var_inter, 'BIC')
3 modeloForwAIC = glm_backward(y_train, x_train, var_cont, var_categ, var_inter, 'AIC')
4 modeloForwBIC = glm_backward(y_train, x_train, var_cont, var_categ, var_inter, 'AIC')
5 modeloStepAIC = glm_backward(y_train, x_train, var_cont, var_categ, var_inter, 'AIC')
6 modeloStepBIC = glm_backward(y_train, x_train, var_cont, var_categ, var_inter, 'AIC')

```

Método	Métrica	Pseudo R ² Train	Pseudo R ² Test	Nº Parám.	AUC
Backward	AIC	0.2938113519165	0.2875414202159	75	0.8407524963023
Backward	BIC	0.2938113519165	0.2875414202159	75	0.8407524963023
Forward	AIC	0.2959551087774	0.3029646817758	79	0.8456687774041
Forward	BIC	0.2959551087774	0.3029646817758	79	0.8456687774041
Stepwise	AIC	0.2959353605732	0.3029032653698	79	0.8456757956497
Stepwise	BIC	0.2959353605732	0.3029032653698	79	0.8456757956497

Table 9.1: Comparación de métodos de selección clásica de variables, con interacciones.



A la vista de los resultados, elegiría el modelo Backward con BIC, ya que logra un buen equilibrio entre simplicidad y rendimiento. Este modelo utiliza 75 parámetros, el menor número entre las opciones disponibles, y aún así alcanza un Pseudo R² en prueba competitivo (0.2875) y un AUC razonablemente alto (0.8407). Aunque Forward y Stepwise con BIC logran un rendimiento ligeramente superior (Pseudo R² de 0.3029 y AUC de 0.8456), lo hacen a costa de incluir más parámetros (79), lo que puede incrementar la complejidad y el riesgo de sobreajuste. Por tanto, prefiero Backward con BIC por ser más parsimonioso y eficiente.

Ahora, al igual que en la regresión lineal, implemento la **selección de variables aleatorias**, donde se realizan múltiples iteraciones para identificar las combinaciones más relevantes de variables en cada división aleatoria del conjunto de datos de entrenamiento. Las variables seleccionadas y sus respectivas fórmulas se registran, y al finalizar las iteraciones, se calcula la frecuencia de cada fórmula para identificar los modelos más consistentes y representativos.

```

1 # Inicializo un diccionario para almacenar las fórmulas y variables seleccionadas.
2 variables_seleccionadas = {
3     'Formula': [],
4     'Variables': []}
5
6 # Realizo 20 iteraciones de selección aleatoria.
7 for x in range(20):
8     print('-----iter:_' + str(x))
9     # Divido los datos de entrenamiento en conjuntos de entrenamiento y prueba.
10    x_train2, x_test2, y_train2, y_test2 = train_test_split(x_train, y_train, test_size =
11        0.3, random_state = 1234567 + x)
12    # Realizo la selección stepwise utilizando el criterio BIC en la submuestra.
13    modelo = glm_backward(y_train2.astype(int), x_train2, var_cont, var_categ, var_inter, '
14        BIC')
15    # Almaceno las variables seleccionadas y la fórmula correspondiente.
16    variables_seleccionadas['Variables'].append(modelo['Variables'])
17    variables_seleccionadas['Formula'].append(sorted(modelo['Modelo'].feature_names_in_))
18
19 # Uno las variables en las fórmulas seleccionadas en una sola cadena.
20 variables_seleccionadas['Formula'] = list(map(lambda x: '+'.join(x), variables_seleccionadas[
21     'Formula']))
22
23 # Calculo la frecuencia de cada fórmula y ordenarlas por frecuencia.
24 frecuencias = Counter(variables_seleccionadas['Formula'])
25 frec_ordenada = pd.DataFrame(list(frecuencias.items()), columns = ['Formula', 'Frecuencia'])
26 frec_ordenada = frec_ordenada.sort_values('Frecuencia', ascending = False).reset_index()
27
28 # Identifico las dos modelos más frecuentes y las variables correspondientes.
29 var_1 = variables_seleccionadas['Variables'][variables_seleccionadas['Formula'].index(
30     frec_ordenada['Formula'][0])]
31 var_2 = variables_seleccionadas['Variables'][variables_seleccionadas['Formula'].index(
32     frec_ordenada['Formula'][1])]
33 var_3 = variables_seleccionadas['Variables'][variables_seleccionadas['Formula'].index(
34     frec_ordenada['Formula'][2])]

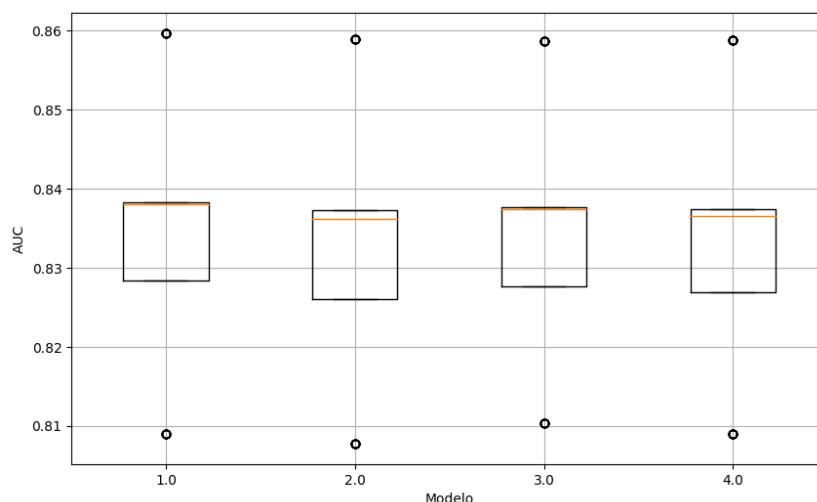
```

Seguidamente, realizo una comparación final de modelos utilizando **validación cruzada** para evaluar el desempeño de cuatro candidatos: el modelo ganador previo y los tres modelos más frecuentes obtenidos mediante selección aleatoria. Para cada modelo, calculo el AUC a través de validación cruzada con 5 particiones, repitiendo el proceso 20 veces. Los resultados se consolidan en un DataFrame y se visualizan mediante un boxplot, permitiendo identificar el modelo con mejor desempeño promedio y mayor estabilidad.

```

1 results = pd.DataFrame({
2     'AUC': [],
3     'Resample': [],
4     'Modelo': []})
5 for rep in range(20):
6     modelo1 = validacion_cruzada_glm(5, x_train, y_train, modeloForwBIC['Variables']['cont'],
7         modeloForwBIC['Variables']['categ'], modeloForwBIC['Variables']['inter'])
8     modelo2 = validacion_cruzada_glm(5, x_train, y_train, var_1['cont'], var_1['categ'],
9         var_1['inter'])
10    modelo3 = validacion_cruzada_glm(5, x_train, y_train, var_2['cont'], var_2['categ'],
11        var_2['inter'])
12    modelo4 = validacion_cruzada_glm(5, x_train, y_train, var_3['cont'], var_3['categ'],
13        var_3['inter'])
14    results_rep = pd.DataFrame({
15        'AUC': modelo1 + modelo2 + modelo3 + modelo4
16        , 'Resample': ['Rep' + str((rep + 1))]*5*4
17        , 'Modelo': [1]*5 + [2]*5 + [3]*5 + [4]*5})
18    results = pd.concat([results, results_rep], axis = 0)
19
20 # Boxplot de la validacion cruzada
21 plt.figure(figsize=(10, 6))
22 plt.grid(True)
23 grupo_metrica = results.groupby('Modelo')['AUC']
24 boxplot_data = [grupo_metrica.get_group(grupo).tolist() for grupo in grupo_metrica.groups]
25 plt.boxplot(boxplot_data, labels=grupo_metrica.groups.keys())
26 plt.xlabel('Modelo')
27 plt.ylabel('AUC')
28 plt.show()

```



```
1 # Calculo la media del AUC por modelo
2 media_auc = results.groupby('Modelo')['AUC'].mean()
3 print(media_auc)
```

Out[]: 1.0 → 0.834664, 2.0 → 0.833244, 3.0 → 0.834343, 4.0 → 0.833752

```
1 # Calcular la desviación estándar del AUC por modelo
2 std_auc = results.groupby('Modelo')['AUC'].std(std_auc)
3 print(std_auc)
```

Out[]: 1.0 → 0.016528, 2.0 → 0.016721, 3.0 → 0.015782, 4.0 → 0.016279

```
1 # Cuento el número de parámetros en cada modelo
2 num_params = [len(modeloForwBIC['Modelo'].coef_[0]), len(frec_ordenada['Formula'][0].split('+'))
3               , len(frec_ordenada['Formula'][1].split('+')), len(frec_ordenada['Formula'][2].split('+'))]
4 print(num_params)
```

Out[]: 1.0 → 75, 2.0 → 72, 3.0 → 75, 4.0 → 74

Considero que el mejor modelo es el modelo 3 pues tiene el mayor promedio de AUC (0.834343) y además una desviación estándar baja (0.015782), lo cual indica un buen rendimiento y consistencia. Aunque el modelo 1 tiene una desviación estándar ligeramente más baja (0.016528), su AUC promedio es menor (0.834664 frente a 0.834343), y no hay una diferencia significativa en simplicidad. Por lo tanto, el modelo 3 parece ser el mejor porque logra un excelente desempeño y consistencia, mientras mantiene la simplicidad necesaria (con 75 parámetros en total).

```
1 ModeloGanador=glm_backward(y_train,x_train,var_2['cont'],var_2['categ'],var_2['inter'],'BIC')
```

A continuación, realizo una evaluación detallada del modelo final para asegurar su desempeño y robustez. Primero, genero una rejilla de puntos de corte, desde 0 a 1 con incrementos de 0.01, y calculo las métricas principales para cada corte: precisión (accuracy), sensibilidad, especificidad, valor predictivo positivo y negativo. También incluyo el índice de Youden, que combina sensibilidad y especificidad, y visualizo los resultados en gráficos que muestran cómo varían Youden y la precisión a lo largo de los cortes. Identifico los mejores puntos de corte, y comparo su desempeño.

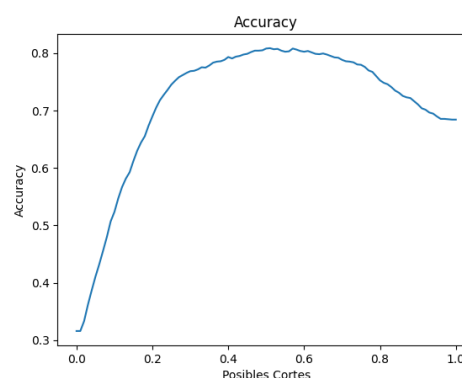
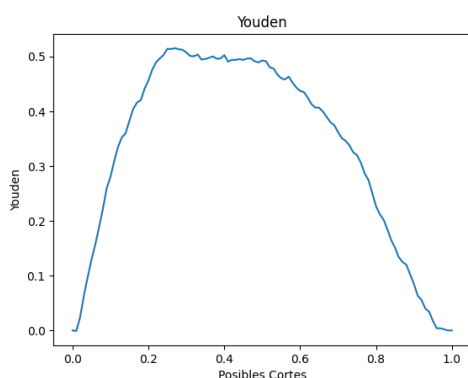
```
1 # Generamos una rejilla de puntos de corte de 0 a 1 con intervalo de 0.01
2 posiblesCortes = list(np.arange(0, 1.01, 0.01))
3
4 # Creamos un DataFrame para almacenar las métricas para cada punto de corte
5 rejilla = pd.DataFrame({
6     'PtoCorte': [],
7     'Accuracy': [],
8     'Sensitivity': [],
9     'Specificity': [],
10    'PosPredValue': [],
11    'NegPredValue': []})
```



```

12 # Calculamos las métricas para el punto de corte actual y lo agregamos al DataFrame
13 for pto_corte in posiblesCortes:
14     rejilla = pd.concat([rejilla, sensEspCorte(ModeloGanador['Modelo'], x_test, y_test,
15         pto_corte, var_2['cont'], var_2['categ'], var_2['inter'])], axis=0)
16
17 rejilla['Youden'] = rejilla['Sensitivity'] + rejilla['Specificity'] - 1 # Calculamos el
18     índice de Youden
19 # Reindexamos el DataFrame para que los índices sean consecutivos
20 rejilla.index = list(range(len(rejilla)))
21
22 plt.plot(rejilla['PtoCorte'], rejilla['Youden'])
23 plt.xlabel('Posibles_Cortes')
24 plt.ylabel('Youden')
25 plt.title('Youden')
26 plt.show()
27
28 plt.plot(rejilla['PtoCorte'], rejilla['Accuracy'])
29 plt.xlabel('Posibles_Cortes')
30 plt.ylabel('Accuracy')
31 plt.title('Accuracy')
32 plt.show()

```



```

1 # Encuentro el punto de corte que maximiza el índice de Youden
2 p1 = rejilla['PtoCorte'][rejilla['Youden'].idxmax()]
3 # Encuentro el punto de corte que maximiza la precisión ( Accuracy )
4 p2 = rejilla['PtoCorte'][rejilla['Accuracy'].idxmax()]

```

El resultado es 0.27 para youden y 0.51 para Accuracy. Procedo a compararlos.

```

1 sensEspCorte(ModeloGanador['Modelo'], x_test, y_test, p1, var_3['cont'], var_3['categ'],
2     var_3['inter'])

```

PtoCorte	Accuracy	Sensitivity	Specificity	PosPredValue	NegPredValue
0.27	0.758005	0.756335	0.758776	0.591463	0.870868

```

1 sensEspCorte(ModeloGanador['Modelo'], x_test, y_test, p2, var_2['cont'], var_2['categ'],
2     var_2['inter'])

```

PtoCorte	Accuracy	Sensitivity	Specificity	PosPredValue	NegPredValue
0.51	0.808498	0.575049	0.916292	0.760309	0.823625

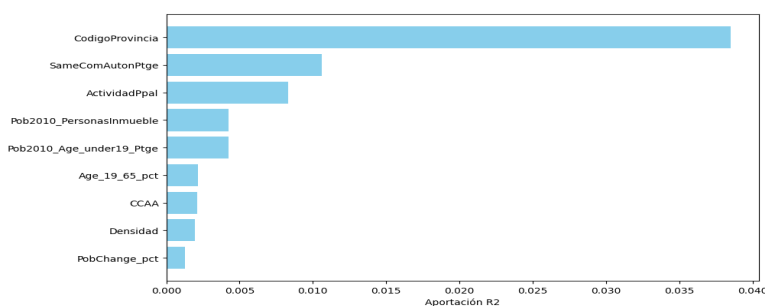
He elegido el punto de corte de 0.27 porque prioriza la sensibilidad (0.756335), lo cual es crucial en el contexto de las elecciones. Mi objetivo principal es identificar la mayor cantidad posible de personas que probablemente se abstendrán, ya que estas son el público objetivo para diseñar e implementar estrategias que aumenten la participación electoral. Aunque esto puede generar un mayor número de falsos positivos (personas identificadas como posibles abstencionistas pero que no lo son), este error es más aceptable que el de los falsos negativos, ya que no intervenir sobre personas que realmente se abstendrán tendría un impacto más negativo en la tasa de participación.

Además, el punto de corte 0.27 mantiene un equilibrio razonable, con una especificidad de 0.758776, lo que significa que aún hay una buena capacidad para filtrar aquellos que no se abstendrán. Este balance asegura que las intervenciones sean lo suficientemente amplias para maximizar su impacto, sin ser completamente ineficientes en términos de asignación de recursos. Por lo tanto, este punto de corte es adecuado para abordar el objetivo principal del modelo: reducir la abstención.

Por último, analizo las variables más importantes del modelo ganador con `impVariablesLog` y reviso los coeficientes del modelo para interpretar el impacto de cada variable.

```
1 # Veo las variables mas importantes del modelo ganador
2 impVariablesLog(ModeloGanador['Modelo'], y_train, x_train, var_2['cont'], var_2['categ'],
  var_2['inter'])
```

	Variables	R2
2	PobChange_pct	0.001266
6	Densidad	0.001974
4	CCAA	0.002104
0	Age_19_65_pct	0.002170
7	Pob2010_Age_under19_Ptge	0.004220
8	Pob2010_PersonasInmueble	0.004265
5	ActividadPpal	0.008293
1	SameComAutonPtge	0.010593
3	CodigoProvincia	0.038493



```
1 # Veo los coeficientes del modelo ganador
2 [language=python]
3 coeficientes = ModeloGanador['Modelo'].coef_
4 nombres_caracteristicas = crear_data_modelo(x_train, var_3['cont'], var_3['categ'], var_3['
  inter']).columns # Suponiendo que X_train es un DataFrame de pandas
5 # Imprime los nombres de las características junto con sus coeficientes
6 for nombre, coef in zip(nombres_caracteristicas, coeficientes[0]):
7     print(f"Variable: {nombre}, Coeficiente: {coef}")
```

Out[]: Variable: Age_19_65_pct, Coeficiente: -0.026230287278811163

Variable: SameComAutonPtge, Coeficiente: -0.03210001586421408

Variable: Densidad_Baja, Coeficiente: -0.6637081637123843

Variable: Densidad_MuyBaja, Coeficiente: -0.22510557949572063 ...

Interpretación de coeficientes:

- Variable continua (SameComAutonPtge): Tiene un coeficiente de -0.0321, lo que implica que por cada aumento del 1% en el porcentaje de personas nacidas en la misma comunidad autónoma, la probabilidad de abstención disminuye en un 0.032%, manteniendo constantes las demás variables. Este resultado sugiere que un mayor sentido de arraigo territorial está asociado con una menor tendencia a la abstención electoral.
- Variable binaria (Densidad_Baja): Su coeficiente es -0.6637, lo que indica que los municipios con baja densidad de población tienen una probabilidad de abstención promedio 0.664% menor, manteniendo constantes las demás variables. Esto podría reflejar un efecto en el que comunidades más pequeñas o rurales tienden a estar más comprometidas políticamente.

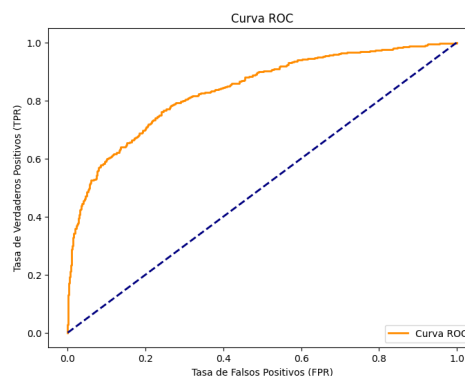
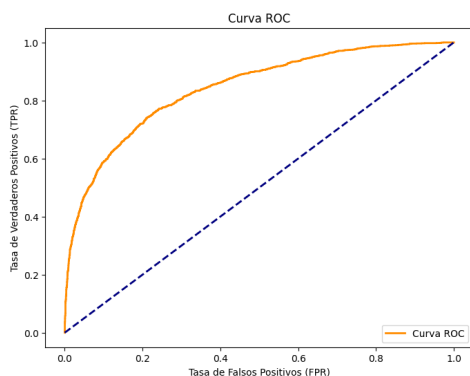
Evalúo la estabilidad del modelo calculando el pseudo R^2 en los conjuntos de entrenamiento y prueba, confirmando que las diferencias son mínimas y que el modelo es robusto.

```
1 # Evaluamos la estabilidad del modelo a partir de las diferencias en train y test:
2 pseudoR2(ModeloGanador['Modelo'], ModeloGanador['X'], y_train) #0.2936070417424089
3 x_test_ModeloGanador = crear_data_modelo(x_test, ModeloGanador['Variables']['cont'],
  ModeloGanador['Variables']['categ'], ModeloGanador['Variables']['inter'])
4 pseudoR2(ModeloGanador['Modelo'], x_test_ModeloGanador, y_test) #0.2820159405422987
```

Es poca la diferencia, por lo que el modelo se puede considerar robusto

Finalmente, comparo el área bajo la curva ROC y las medidas de calidad entre los datos de entrenamiento y prueba, incluyendo sensibilidad y especificidad en el punto de corte 0.5, verificando que las métricas son consistentes en ambos conjuntos, lo que reafirma la calidad del modelo ajustado.

```
1 # Calculamos la diferencia del Area bajo la curva ROC en train y test
2 curva_roc(crear_data_modelo(x_train , var_3['cont'], var_3['categ'], var_3['inter'], y_train
  , ModeloGanador)) #Área bajo la curva ROC = 0.8419930052870627
3 curva_roc(x_test_ModeloGanador, y_test, ModeloGanador) #Área bajo la curva ROC =
  0.8366029585414682
```



```
1 # Calculamos la diferencia de las medidas de calidad entre train y test
2 sensEspCorte(ModeloGanador['Modelo'], x_train, y_train, 0.27, var_2['cont'], var_2['categ'],
  var_2['inter'])
```

PtoCorte	Accuracy	Sensitivity	Specificity	PosPredValue	NegPredValue
0.27	0.765594	0.76155	0.767411	0.59534	0.877489

```
1 sensEspCorte(ModeloGanador['Modelo'], x_test, y_test, 0.27, var_2['cont'], var_2['categ'],
  var_2['inter'])
```

PtoCorte	Accuracy	Sensitivity	Specificity	PosPredValue	NegPredValue
0.27	0.758005	0.756335	0.758776	0.591463	0.870868

Con base en los resultados, considero que el modelo de regresión logística es sólido y confiable para predecir municipios con una abstención alta (superior al 30%). El área bajo la curva ROC, consistente entre entrenamiento (0.8419) y prueba (0.8366), demuestra una alta capacidad discriminativa del modelo. Además, las métricas de sensibilidad (76%) y especificidad (75%) en el punto de corte 0.27 aseguran un buen equilibrio entre identificar correctamente municipios con alta abstención y evitar clasificaciones erróneas. La estabilidad de estas métricas entre los conjuntos de entrenamiento y prueba refuerza la validez del modelo, garantizando que no está sobreajustado y que generaliza bien a datos no vistos. Esto me permite confiar en el modelo para priorizar municipios con alto riesgo de abstención y enfocar esfuerzos en reducirla de manera eficiente.