UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE ESTUDIOS ESTADÍSTICOS

TAREA EVALUABLE

MÓDULO: MINERÍA DE DATOS Y MODELIZACIÓN PREDICTIVA - PREPARACIÓN DE DATOS Y REGRESIONES

Jorge González Perea 51553561G

Máster en Big Data, Data Science & Inteligencia Artificial

Curso académico 2024-2025

Índice

1. Introducción.

En este proyecto se trabaja con una base de datos que incluye información demográfica sobre las últimas elecciones en España. Se tienen 8117 filas, correspondientes a distintos municipios, y 41 columnas, que contienen los datos de las variables. De las 41 variables, 7 de ellas son de interés:

- **AbstentionPtge**: porcentaje de abstención.
- **Izda_Pct**: porcentaje de votos a partidos de izquierda
- **Dcha_Pct**: porcentaje de votos a partidos de derecha
- **Otros_Pct**: porcentaje de votos a partidos distintos de PP, Ciudadanos, PSOE y Podemos.
- **AbstencionAlta**: variable dicotómica que toma el valor 1 si el porcentaje de abstención es superior al 30 % y, 0, en otro caso.
- **Izquierda**: variable dicotómica que toma el valor 1 si la suma de los votos de izquierdas es superior a la de derechas y otros y, 0, en otro caso.
- **Derecha**: variable dicotómica que toma el valor 1 si la suma de los votos de derecha es superior a la de izquierda y otros y, 0, en otro caso.

Para este estudio se han elegido dos variables objetivo de las siete anteriores. Una de ellas (numérica) se someterá a un modelo de regresión lineal para su correspondiente predicción. Con la otra (dicotómica) se hará lo mismo pero con un modelo de regresión logística. Las variables escogidas son **Izda Pct** e **Izquierda**. Las cinco variables restantes son eliminadas de la base de datos debido a que no se utilizarán como variables explicativas:

Además de estas, también se tienen diversas variables en la base de datos. Algunas son de carácter geográfico (nombre del municipio, código de provincia, CCAA, etc.), otras son datos demográficos (población, porcentaje de nativos de cada CCAA y extranjeros, datos sobre empleo en diferentes sectores, etc.) y otras están relacionadas con los resultados de las elecciones.

El objetivo es depurar esta base de datos y, posteriormente, crear un modelo de regresión lineal y otro de regresión logística que permitan predecir variables continuas y resolver problemas de clasificación respectivamente para las dos variables objetivo elegidas. El resultado esperado de este proceso es un análisis sobre los votos a partidos de izquierda a nivel municipal para poder observar tendencias y poder hacer predicciones.

2. Depuración de datos.

A continuación se encuentra un listado con todas las librerías y funciones importadas.

```
import os
import pandas as pd
import numpy as np
import pickle
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from itertools import combinations
from collections import Counter
from FuncionesMineria_original import analizar_variables_categoricas,
atipicosAmissing, patron_perdidos, ImputacionCuant, ImputacionCuali,
graficoVcramer, Vcramer, mosaico_targetbinaria, boxplot_targetbinaria,
hist_targetbinaria, hist_target_categorica, lm, Rsq, validacion_cruzada_lm,
modelEffectSizes, crear_data_modelo, Transf_Auto, Rsq, lm, lm_forward,
lm_backward, lm_stepwise, validacion_cruzada_lm, crear_data_modelo
os.chdir(r'C:\Users\jorge\Desktop\M ster\M dulo 7 - Miner a de datos y
modelizaci n predictiva\Parte 1\Tarea')
```

Listing 1: Librerías y funciones.

Para leer los datos desde el archivo **DatosEleccionesEspaña.xlsx** con Python es necesario ejecutar las siguientes instrucciones:

```
datos = pd.read_excel('DatosEleccionesEspa a.xlsx')
datos = datos.drop(columns = ['AbstentionPtge', 'Dcha_Pct', 'Otros_Pct', '
    AbstencionAlta', 'Derecha'])
datos.dtypes
```

Listing 2: Lectura de datos.

Gracias a esta instrucción se genera una variable llamada '**datos**' que tiene formato de **DataFrame** de la librería Pandas. En esta tabla se pueden visualizar algunos de los valores que toman las distintas variables:

Índice	Name	digoProvin	CCAA	Population	otalCensus	stentionPt	stencionAl	Izda_Pct	Dcha_Pct	Otros_Pct
0	Abadía	10	Extremadura	336	282	20.213	0	60.444	35.555	1.778
1	Abertura	10	Extremadura	429	364	25.275	0	54.779	44.118	0.368
2	Acebo	10	Extremadura	569	569	27.241	0	44.203	53.14	0.966
3	Acehúche	10	Extremadura	822	704	30.114	1	50.813	45.325	0
4	Aceituna	10	Extremadura	623	540	30.185	1	44.562	49.867	0.796
5	Ahigal	10	Extremadura	1421	1263	22.565	0	53.068	45.501	0.409
6	Alagón del Río	10	Extremadura	923	826	30.024	1	59.343	37.89	0.692
7	Albalá	10	Extremadura	730	629	27.027	0	37.037	60.567	0.218
8	Alcántara	10	Extremadura	1571	1320	22.803	0	46.81	50.736	0.785
9	Alcollarín	10	Extremadura	256	266	24.436	0	37.811	58.209	0.498
10	Alcuéscar	10	Extremadura	2759	2290	40.524	1	55.286	41.923	0.734
11	Aldea del Cano	10	Extremadura	667	595	21.681	0	61.159	37.339	0
12	Aldeacentenera	10	Extremadura	646	560	18.214	0	57.861	38.21	0.655

Figura 1: Algunas columnas de la base de datos sin depurar.

Con el comando **datos.dtypes** se imprime por pantalla el nombre de todas las columnas de la tabla 1 junto con el formato de datos que contiene cada una (que tiene que ser único según el funcionamiento de un DataFrame). En la impresión por pantalla se puede ver una serie de variables que vienen en formato flotante (**float64**) o entero (**int64**) y que deberían ser objetos o cadenas de caracteres:

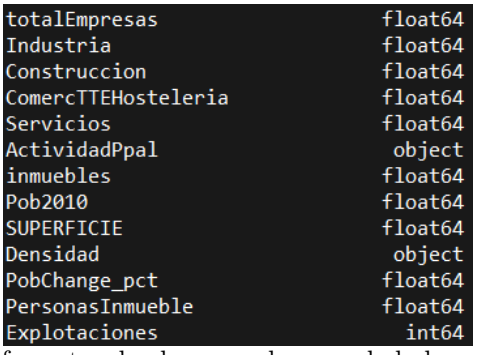

Figura 2: formatos de algunas columnas de la base de datos.

En el siguiente listado se encuentra el código para dividir las variables en función de su tipo (numéricas o categóricas) así como la creación de una tabla que contiene distintas medidas estadísticas relativas a las variables y que pueden ser de ayuda a la hora de llevar a cabo una depuración de los datos. Las variables **'CodigoProvincia** e **Izquierda** se pueden convertir a categóricas debido al reducido conjunto de sus valores.

```
numericasAcategoricas = ['CodigoProvincia','Izquierda']
for var in numericasAcategoricas:
    datos[var] = datos[var].astype(str)
variables = datos.columns.tolist()
numericas = datos.select_dtypes(include=['int', 'int32', 'int64','float',
                    'float32', 'float64']).columns.tolist()
categoricas = [variable for variable in variables if variable not in numericas]
descriptivos_num = datos.describe().T
for num in numericas:
    descriptivos_num.loc[num, "Asimetria"] = datos[num].skew()
    descriptivos_num.loc[num, "Kurtosis"] = datos[num].kurtosis()
    descriptivos_num.loc[num, "Rango"] = np.ptp(datos[num].dropna().values)
```

Listing 3: Separación de variables y análisis estadístico básico.

Y la tabla mencionada incluye una recopilación de medidas estadísticas de relevancia para esta base de datos que permiten localizar valores atípicos, no numéricos, etc. como se verá en el apartado 2.2:

Índice	count	mean	std	min	25%	50%	75%	max	Asimetria	Kurtosis	Rango
CodigoProvincia	8117	26.6647	14.8934	1	13	26	41	50	0.00961447	-1.32302	49
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
Izda_Pct	8117	34.404	16.4843	0	21.891	35.165	46.032	94.117	0.0598817	-0.493538	94.117
Izquierda	8117	0.222866	0.416194	0	0	0	0	1	1.33208	-0.225612	1
Age_0-4_Ptge	8117	3.01827	2.05263	0	1.389	2.975	4.533	13.245	0.343639	-0.206688	13.245
Age_under19_Ptge	8117	13.5641	6.77745	0	8.334	13.881	19.055	33.696	-0.104763	-0.79225	33.696
Age_19_65_pct	8117	57.3706	6.81864	23.459	53.845	58.655	61.818	100.002	-0.814264	2.15584	76.543
Age_over65_pct	8117	29.0653	11.767	-18.052	19.827	27.559	36.911	76.472	0.584708	0.102323	94.524
WomanPopulationPtge	8117	47.3023	4.36235	11.765	45.725	48.485	50	72.683	-1.6711	5.80063	60.918
ForeignersPtge	8117	5.61832	7.3487	-8.96	1.06	3.59	8.18	71.47	2.49826	11.3568	80.43
SameComAutonPtge	8117	81.6335	12.2873	0	75.806	84.493	90.462	127.156	-1.52276	3.47954	127.156
SameComAutonDiffProvPtge	8117	4.33764	6.39494	0	0.676	2.19	5.277	67.308	3.28683	14.5601	67.308

Figura 3: Algunas medidas estadísticas de la base de datos sin depurar.

2.1. Corrección de variables categóricas.

El conjunto de variables categóricas está formado por **Name**, **CodigoProvincia**, **CCAA**, **Izquierda**, **ActividadPpal** y **Densidad**. Mediante el comando **analizar_variables_categoricas** se pueden visualizar todos los posibles valores de cada columna.

```
frecuencias = analizar_variables_categoricas(datos)
frecuencias

# Cambio de '?' por nan en la columna 'Densidad'.
datos['Densidad'] = datos['Densidad'].replace('?', np.nan)
analizar_variables_categoricas(datos)

# Cambio de cadenas de caracteres 'nan' a NaN.
for x in categoricas:
    datos[x] = datos[x].replace('nan', np.nan)
datos[variables].isna().sum()

# Agrupaci n de categor as poco representadas:
datos['ActividadPpal'] = datos['ActividadPpal'].replace({'Otro':'Otro', '
ComercTTEHosteleria': 'ComercioyHosteleria', 'Servicios': '
Servicios_Construccion_Industria', 'Construccion': '
Servicios_Construccion_Industria', 'Industria': '
Servicios_Construccion_Industria'})
```

Listing 4: Comprobación de variables categóricas.

Este código proporciona la siguiente salida:

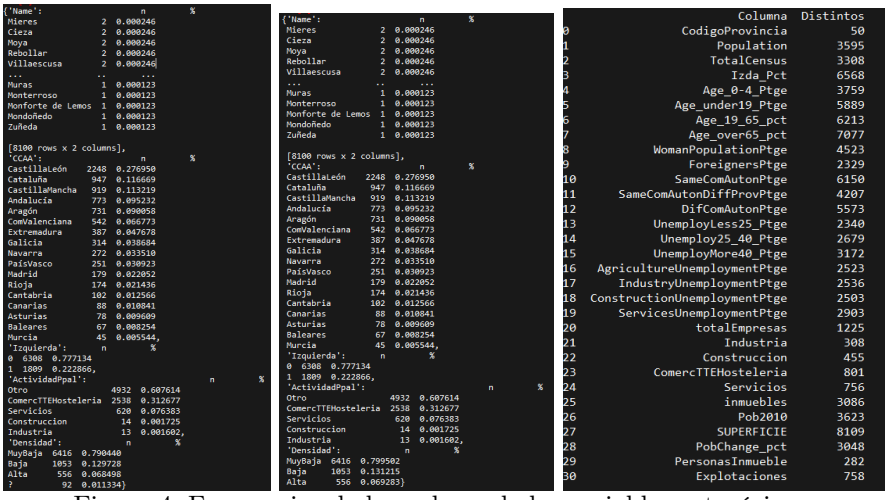

Figura 4: Frecuencias de los valores de las variables categóricas.

En el listado 4 se llevan a cabo los siguientes procesos:

- Se comprueba la frecuencia de cada valor en las columnas categóricas.
- Se sustituyen los caracteres '?' de la columna **Densidad** por **NaN**.
- Se sustituyen posibles valores **NaN** que estén escritos como la cadena de caracteres 'nan' por el valor **NaN** real.
- Se agrupan las categorías poco representadas de la columna **ActividadPpal** para evitar errores en la codificación de los datos más adelante.

Los municipios repetidos podrían estar en la misma CCAA, lo cual sería un error evidente. Con ayuda de la función **analizar_variables_categoricas** de la librería **FuncionesMineria.py** se obtienen las frecuencias de cada municipio. Se observa que 17 de ellos se repiten y se comprueba si es un error o no. Ninguno de los municipios está repetido, ya que los 17 que se repiten dos veces (no hay niguno que se repita 3 veces o más) están en CCAA diferentes.

Nota: en la descripción de esta variable se indica que simboliza la densidad de habitantes por hectárea (d), y que su valor puede comprenderse entre 3 posibles: 'MuyBaja' si d <1 hab/ha, 'Baja' si 1 <d <5, y 'Alta' si d >5 hab/ha. Por lo tanto, se podría calcular la densidad como el cociente de los datos de la columna **Population** entre los de la columna **SUPERFICIE** y asignar una de estas cadenas de caracteres en función del valor numérico para cada municipio de forma muy sencilla. Sin embargo, se desconocen las unidades de la variable **SUPERFICIE** y también se desconoce qué censo poblacional se ha utilizado para calcular los valores que sí están disponibles en la columna **Densidad**, luego no es conveniente hacerlo y es preferible tratar esos valores como *missing*.

2.2. Corrección de datos numéricos.

El siguiente paso consiste en corregir los valores atípicos, porcentajes erróneos, etc. Para ello es necesario una lectura manual de los datos con ayuda de, por ejemplo, la tabla de la figura 3.

Los errores detectados y sus soluciones son los siguientes:

- En la columna **Explotaciones** hay valores atípicos que se parecen haber sido acotados en 9999, por lo que también se sustituyen por **NaN**.
- Algunos porcentajes están mal indicados, pues su valor está fuera del intervalo [0, 100]; y ninguno representa un decrecimiento numérico (<0 %) o representan algo que pueda exceder el total de la población de forma coherente (>100 %). Para solucionarlo se sustituyen todos los valores fuera del rango por **NaN**.

A continuación se encuentran las instrucciones para corregir estos errores:

```
# Missings no declarados variables cuantitativas (-1, 99999)
datos['Explotaciones'] = datos['Explotaciones'].replace(99999, np.nan)

# Valores fuera de rango. 'PobChange_pct' puede contener porcentajes negativos.
c_no_porcentajes = ['Name', 'CodigoProvincia', ..., 'Explotaciones']
c_porcentajes = [c for c in variables if c not in c_no_porcentajes]
for var in c_porcentajes:
    if var != 'PobChange_pct':
        datos[var] = [x if 0 <= x <= 100 else np.nan for x in datos[var]]

```

Listing 5: Corrección de errores en variables numéricas.

2.3. Valores atípicos y *missing*. Imputaciones.

Antes de proceder, conviene definir las variables objetivo y diferenciar todas las columnas:

```
        datos = datos.set_index(datos['Name']).drop('Name', axis = 1)
        varObjCont = datos['Izda_Pct']
        varObjBin = datos['Izquierda']
        datos_input = datos.drop(['Izda_Pct', 'Izquierda'], axis = 1)
        variables_input = list(datos_input.columns)

        numericas_input = datos_input.select_dtypes(include = ['int', 'int32', '
    int64','float', 'float32', 'float64']).columns
        categoricas_input = [variable for variable in variables_input if variable
    not in numericas_input]

```

Listing 6: Definición de variables objetivo e input.

Para el tratamiento de los valores atípicos es necesario calcular la proporción de estos en cada variable, de forma que se puedan identificar aquellas columnas con porcentajes destacables (>50 %).

```
    numAtipicos = {x: atipicosAmissing(datos_input[x])[1] / len(datos_input) for x
     in numericas_input}
    for x in numericas_input:
        datos_input[x] = atipicosAmissing(datos_input[x])[0]


```

Listing 7: Corrección de errores en variables numéricas.

El último paso antes de las imputaciones consiste en tratar las variables con un porcentaje notable de datos **NaN**. En concreto, se calcula la proporción de ellos en cada una y si este es superior al 50 % se elimina la variable de la BBDD, tal y como se puede observar en la imagen **??**.

```
    # Mapa de calor que muestra la matriz de correlaci n:
    patron_perdidos(datos_input)

    # Proporci n de valores perdidos por cada variable.
    prop_missingsVars = datos_input.isna().sum()/len(datos_input)
    prop_missingsVars
    # N mero de valores perdidos por cada observaci n:
    datos_input['prop_missings'] = datos_input.isna().mean(axis = 1)

    # N mero de valores distintos que tiene la nueva variable.
    len(datos_input['prop_missings'].unique())

    # Se eliminan las observaciones con m s del 50% de datos missing.
    eliminar = datos_input['prop_missings'] > 0.5
    datos_input = datos_input[~eliminar]
    varObjBin = varObjBin[~eliminar]
    varObjCont = varObjCont[~eliminar]

    # Transformaci n de la nueva variable a categ rica:
    datos_input["prop_missings"] = datos_input["prop_missings"].astype(str)

    # Se eliminan las variables con m s del 50% de datos missing.
    eliminar = [prop_missingsVars.index[x] for x in range(len(prop_missingsVars))
    if prop_missingsVars[x] > 0.5]
    datos_input = datos_input.drop(eliminar, axis = 1)

    # Se crean otra vez las listas de variables por si acaso:
    variables_input = list(datos_input.columns)
```

```
    categoricas_input = [variable for variable in variables_input if variable not
    in numericas_input]

```

Listing 8: Corrección de errores en variables numéricas.

Para esta base de datos, no existe ninguna variable con una proporción de valores atípicos superior al 10 %, y tan solo hay dos columnas que se acercan (**Population** y **TotalCensus**), por lo que no se elimina ninguna. Además, se observa que la proporción de valores perdidos tiene 9 valores únicos, por lo que se puede convertir en categórica para su codificación.

Por último, los datos *missing* son rellenados con valores aleatorios introducidos mediante código. Basta con ejecutar las siguientes líneas y el depurado de la base de datos estará completo. Esta nueva base de datos limpia se guarda en el archivo **datosTarea.pickle**, que se empleará para el desarrollo de los modelos de regresión lineal y logística.

```
    # Variables cuantitativas:
    for x in numericas_input:
        datos_input[x] = ImputacionCuant(datos_input[x], 'aleatorio')
    # Variables cualitativas:
    for x in categoricas_input:
        datos_input[x] = ImputacionCuali(datos_input[x], 'aleatorio')

    datos_input.isna().sum()

    datosTarea = pd.concat([varObjBin, varObjCont, datos_input], axis = 1)
    with open('datosTarea.pickle', 'wb') as archivo:
        pickle.dump(datosTarea, archivo)

```

Listing 9: Corrección de errores en variables numéricas.

3. Gráficos y presentación de variables.

Los siguientes apartados incluyen algunos gráficos que permiten visualizar la correlación entre variables, coeficientes relevantes, etc. Para ello, se cargan los datos depurados y se definen las variables objetivo continua y binaria:

```
with open('datosTarea.pickle', 'rb') as f:
    datos = pickle.load(f)

# VARIABLES OBJETIVO.
varObjCont = datos['Izda_Pct']
varObjBin = datos['Izquierda']
datos_input = datos.drop(['Izda_Pct', 'Izquierda'], axis = 1)
variables = list(datos_input.columns)
```

Listing 10: Librerías y lectura de datos.

3.1. Coeficientes V de Cramer.

La V de Cramer indica el grado de asociación entre dos variables categóricas. La función **VCramer** de **FuncionesMineria** permite calcular estos coeficientes. Con el uso de otra función, **graficoVCramer** se genera un gráfico de barras que permite comparar el valor de este coeficiente para todas las variables del DataFrame. El código es el siguiente:

```
# Gr ficos comparando los V de Cramer.
graficoVcramer(datos_input, varObjBin)
graficoVcramer(datos_input, varObjCont)

# C lculo de los coeficientes V de Cramer:
VCramer = pd.DataFrame(columns=['Variable', 'Objetivo', 'Vcramer'])
for variable in variables:
    v_cramer = Vcramer(datos_input[variable], varObjCont)
    VCramer = VCramer.append({'Variable': variable, 'Objetivo': varObjCont.
name, 'Vcramer': v_cramer}, ignore_index=True)
for variable in variables:
    v_cramer = Vcramer(datos_input[variable], varObjBin)
    VCramer = VCramer.append({'Variable': variable, 'Objetivo': varObjBin.name
, 'Vcramer': v_cramer}, ignore_index=True)
```

Listing 11: Código para generar un gráfico y calcular las V de Cramer.

Los bucles de esta celda se emplean para guardar los cálculos en un DataFrame en caso de que sean necesarios más adelante, y se pueden ver en la tabla 7 del anexo. Para estos coeficientes, un valor mayor simboliza una mejor asociación entre variables, mientras que un valor más cercano a 0 tiene significado contrario.

3.2. Variable binaria.

El gráfico obtenido para la variable objetivo binaria (**Izquierda**) es el siguiente:

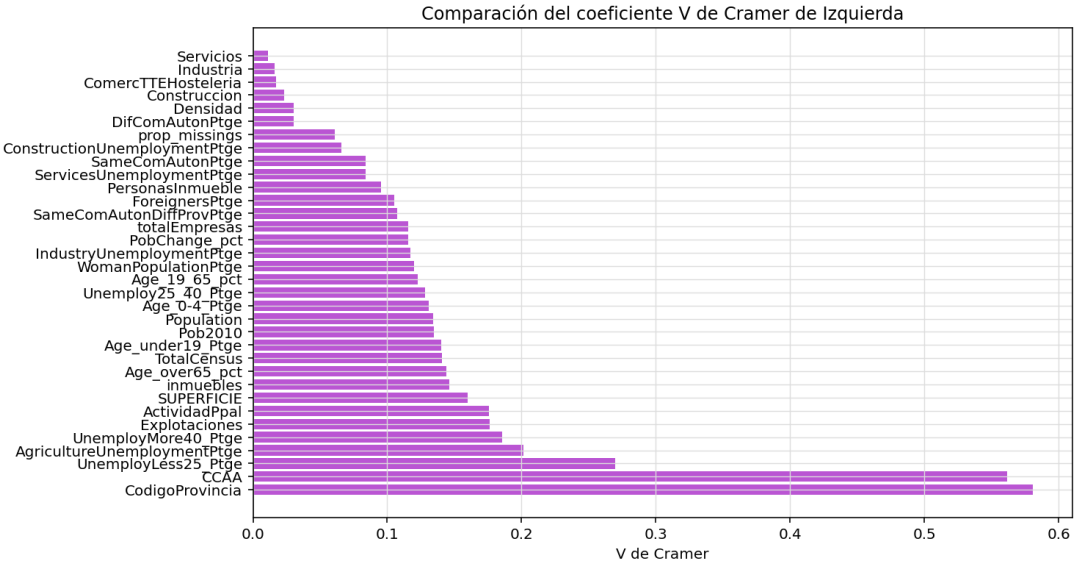

Figura 5: Coeficientes V de Cramer con respecto a la variable binaria **Izquierda**.

Como se puede observar en la imagen 5, las variables categóricas que menos y más se relacionan respectivamente con **Izquierda** son **Densidad** y **CCAA**. Como la variable **Densidad** no provoca mucha variación en los gráficos, se escoge en su lugar **ActividadPpal**, ya que al presentar variación será mejor para predecir la variable objetivo binaria. Los gráficos de mosaico para la relación entre cada una de estas dos variables y la variable objetivo binaria son:

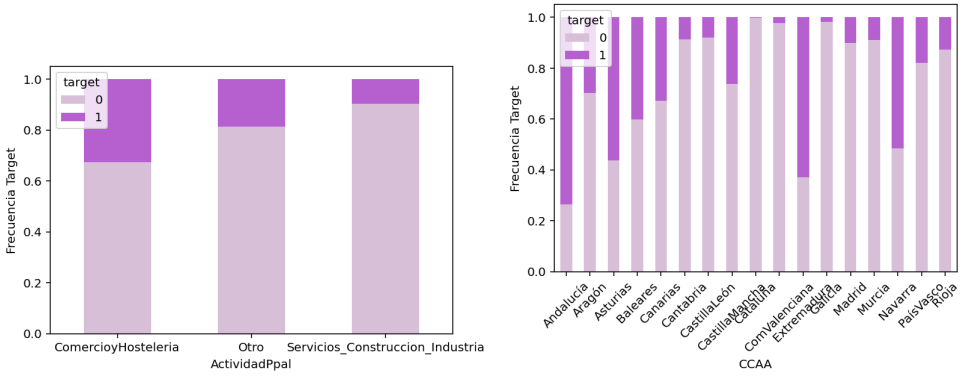

Figura 6: Gráficos de mosaico para relación entre la variable objetivo binaria **Izquierda** con **ActividadPpal** (izqa) y **CCAA** (dcha).

Los gráficos de caja pueden servir para comprobar cómo de bien una variable numérica describe cualquier variable objetivo. Lo que se busca, al igual que con los gráficos de mosaico, son dos variables que muestren diferencias claras entre los dos gráficos de caja para los dos valores 1 y 0 de **Izquierda**. En el caso de la variable objetivo binaria se han elegido las columnas **ServicesUnemploymentPtge** y **UnemployLess25_Ptge**, es decir, los porcentajes de desempleo en el sector de servicios (con coeficiente V de Cramer pequeño según la figura 5), y de desempleo en habitantes menores de 25 años (con V de Cramer alta).

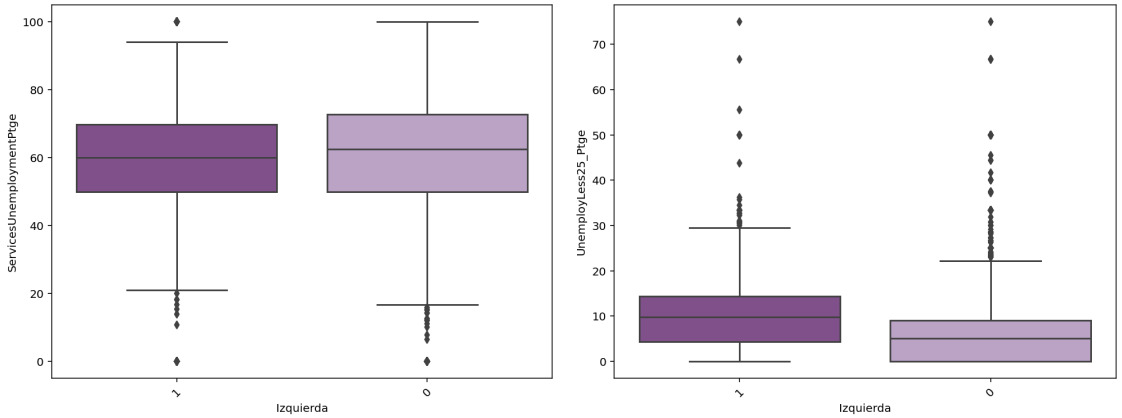

Figura 7: Diagrama de caja para **Izquierda** en función de **ServicesUnemploymentPtge** y **UnemployLess25_Ptge**

Al igual que con los gráficos anteriores, lo que se busca con los histogramas son variables que difieran mucho para poder entrenar el modelo de regresión. En este caso las elegidas son:

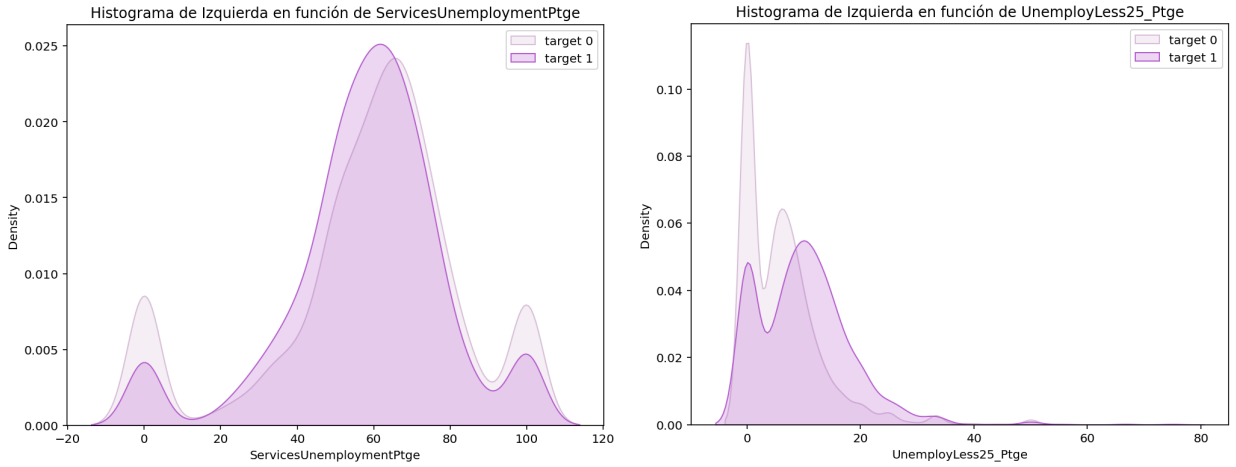

Figura 8: Histograma de **Izquierda** en función de **ServicesUnemploymentPtge** y **UnemployLess25_Ptge**.

3.3. Variable continua.

Por otro lado, para la variable objetivo continua (**Izda_Pct** se genera la siguiente imagen:

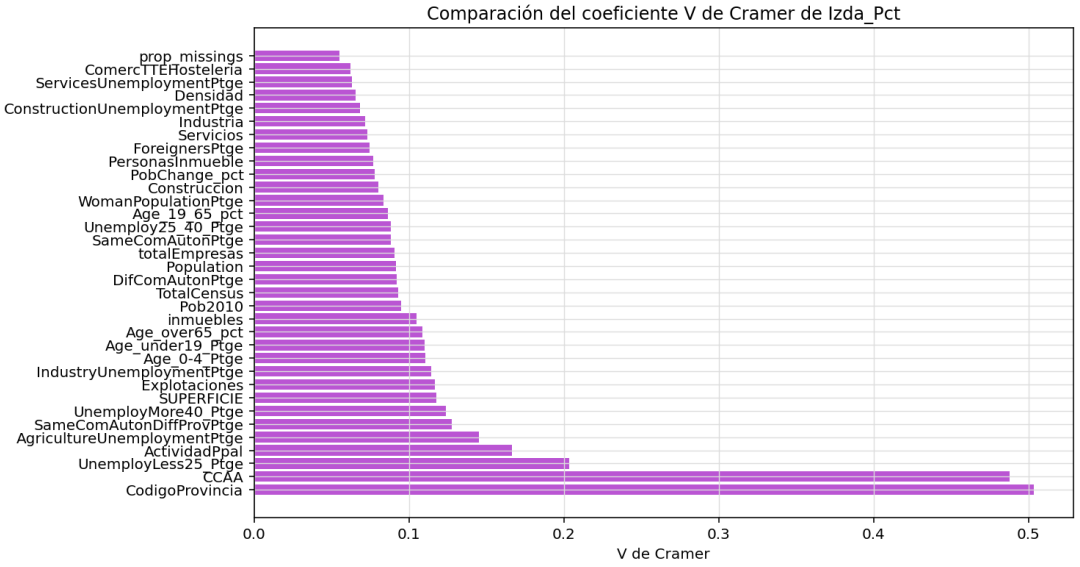

Figura 9: Coeficientes V de Cramer con respecto a la variable continua **Izda_Pct**.

De la misma forma, se buscan variables en la parte alta y baja del gráfico de la figura 9 para poder entrenar el modelo de regresión lineal. En este caso, al tratarse de una variable numérica, se obtienen gráficos con numerosas distribuciones debido a los distintos valores posibles de las otras variables tal y como se ve a continuación.

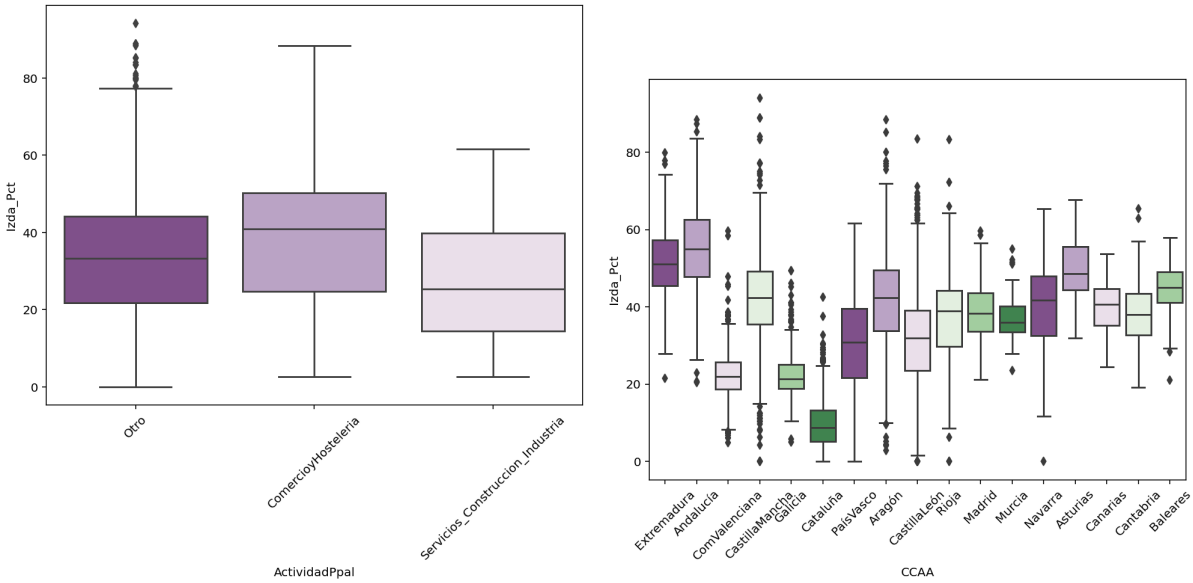

Figura 10: Diagrama de caja para **Izquierda** en función de **ActividadPpal** y **CCAA**.

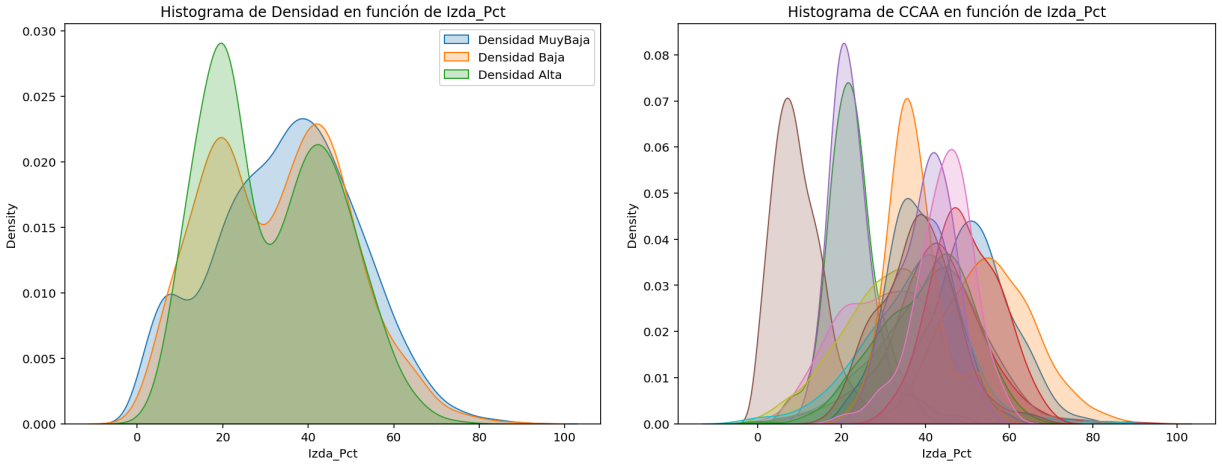

Figura 11: Diagrama de caja para **Izquierda** en función de

Esta matriz nos indica el grado de correlación entre dos variables, lo cual es de gran ayuda para la selección manual.

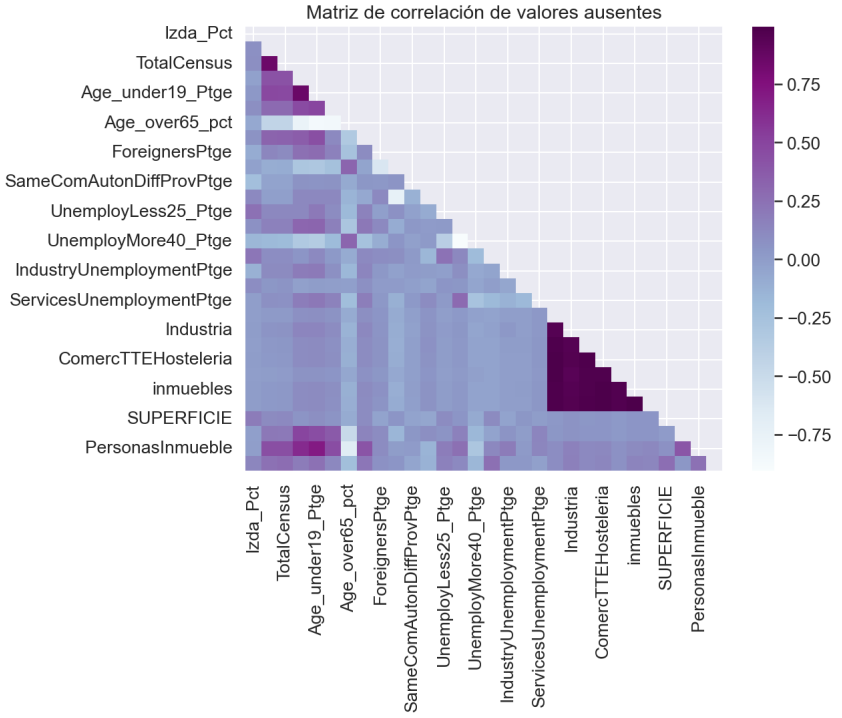

Figura 12: Matriz de correlación.

4. Modelos de regresión lineal.

En este apartado se muestra como crear los modelos de regresión lineal para poder predecir la variable objetivo continua (**Izda_Pct**). Para ello se hace uso de la librería **FuncionesMineria** y de los métodos vistos en clase, como la validación cruzada o los modelos con métricas AIC o BIC.

4.1. Modelo de regresión lineal. Selección de variables clásica.

En este apartado se emplean los métodos *Stepwise selection*, *Backward selection* y *Forward selection*. Estos métodos de selección de variables crean modelos y van añadiendo variables (*Forward*) o quitándolas (*Backward*) del modelo inicial mientras se comprueba la bondad del mismo mediante los criterios AIC (Criterio de Información de Akaike) o BIC (Criterio de Información Bayesiano), que permiten ajustar la complejidad del mismo para evitar seleccionar variables poco útiles o que den información no necesaria. El método *Stepwise* combina los dos anteriores para un desarrollo más eficiente. Lectura de los datos y asignación de variables:

```
with open('datosTarea_cont.pickle', 'rb') as f:
    todo = pickle.load(f)
# Identificacion de variables:
varObjCont = todo['Izda_Pct']
todo = todo.drop('Izda_Pct', axis = 1)
var_cont = todo.select_dtypes(include = ['int', 'float']).columns.tolist()
var_cont_sin_transf = ['CodigoProvincia', ..., 'Explotaciones']
var_categ = [var for var in todo.columns.tolist() if var not in var_cont]
```

Listing 12: Modelos de regresión lineal. Selección de variables.

En este caso se han construido 6 modelos de regresión lineal mediante tres métodos diferentes, cada uno con ambas métricas estudiadas. En este primer caso se han considerado interacciones entre las variables. El código para la creación de los modelos se puede comprobar en el siguiente listado.

```
x_train, x_test, y_train, y_test = train_test_split(todo, varObjCont,
test_size = 0.2, random_state = 1234567)
interacciones  = var_cont_sin_transf
interacciones_unicas = list(combinations(interacciones, 2))

# MODELO STEPWISE AIC.
modeloStepAIC_int = lm_stepwise(y_train, x_train, var_cont_sin_transf,
var_categ, interacciones_unicas, 'AIC')
modeloStepAIC_int['Modelo'].summary()
r_train_stepAIC_int = Rsq(modeloStepAIC_int['Modelo'], y_train,
modeloStepAIC_int['X'])
x_test_modeloStepAIC_int = crear_data_modelo(x_test, modeloStepAIC_int['
Variables']['cont'], modeloStepAIC_int['Variables']['categ'], modeloStepAIC_int
['Variables']['inter'])

r_test_stepAIC_int = Rsq(modeloStepAIC_int['Modelo'], y_test,
x_test_modeloStepAIC_int)
param_stepAIC_int = len(modeloStepAIC_int['Modelo'].params)

# MODELO STEPWISE BIC.
modeloStepBIC_int = lm_stepwise(y_train, x_train, var_cont_sin_transf,
var_categ, interacciones_unicas, 'BIC')
modeloStepBIC_int['Modelo'].summary()
r_train_stepBIC_int = Rsq(modeloStepBIC_int['Modelo'], y_train,
modeloStepBIC_int['X'])
x_test_modeloStepBIC_int = crear_data_modelo(x_test, modeloStepBIC_int['
Variables']['cont'], modeloStepBIC_int['Variables']['categ'], modeloStepBIC_int
['Variables']['inter'])
```

```
r_test_stepBIC_int = Rsq(modeloStepBIC_int['Modelo'], y_test, x_test_modeloStepBIC_int)
param_stepBIC_int = len(modeloStepBIC_int['Modelo'].params)

# MODELO BACKWARD AIC CON EL PRIMER CONJUNTO DE INTERACCIONES.
modeloBackAIC_int = lm_backward(y_train, x_train, var_cont_sin_transf, var_categ, interacciones_unicas, 'AIC')
modeloBackAIC_int['Modelo'].summary()
r_train_backAIC_int = Rsq(modeloBackAIC_int['Modelo'], y_train, modeloBackAIC_int['X'])
x_test_modeloBackAIC_int = crear_data_modelo(x_test, modeloBackAIC_int['Variables']['cont'], modeloBackAIC_int['Variables']['categ'], modeloBackAIC_int['Variables']['inter'])
r_test_backAIC_int = Rsq(modeloBackAIC_int['Modelo'], y_test, x_test_modeloBackAIC_int)
param_backAIC_int = len(modeloBackAIC_int['Modelo'].params)

# MODELO BACKWARD BIC CON EL PRIMER CONJUNTO DE INTERACCIONES.
modeloBackBIC_int = lm_backward(y_train, x_train, var_cont_sin_transf, var_categ, interacciones_unicas, 'BIC')
modeloBackBIC_int['Modelo'].summary()
r_train_backBIC_int = Rsq(modeloBackBIC_int['Modelo'], y_train, modeloBackBIC_int['X'])
x_test_modeloBackBIC_int = crear_data_modelo(x_test, modeloBackBIC_int['Variables']['cont'], modeloBackBIC_int['Variables']['categ'], modeloBackBIC_int['Variables']['inter'])
r_test_backBIC_int = Rsq(modeloBackBIC_int['Modelo'], y_test, x_test_modeloBackBIC_int)
param_backBIC_int = len(modeloBackBIC_int['Modelo'].params)

# MODELO FORWARD AIC CON EL PRIMER CONJUNTO DE INTERACCIONES.
modeloForwAIC_int = lm_stepwise(y_train, x_train, var_cont_sin_transf, var_categ, interacciones_unicas, 'AIC')
modeloForwAIC_int['Modelo'].summary()
r_train_forwAIC_int = Rsq(modeloForwAIC_int['Modelo'], y_train, modeloForwAIC_int['X'])

x_test_modeloForwAIC_int = crear_data_modelo(x_test, modeloForwAIC_int['Variables']['cont'], modeloForwAIC_int['Variables']['categ'], modeloForwAIC_int['Variables']['inter'])
r_test_forwAIC_int = Rsq(modeloForwAIC_int['Modelo'], y_test, x_test_modeloForwAIC_int)
param_forwAIC_int = len(modeloForwAIC_int['Modelo'].params)

# MODELO FORWARD BIC CON EL PRIMER CONJUNTO DE INTERACCIONES.
modeloForwBIC_int = lm_stepwise(y_train, x_train, var_cont_sin_transf, var_categ, interacciones_unicas, 'BIC')
modeloForwBIC_int['Modelo'].summary()
r_train_forwBIC_int = Rsq(modeloForwBIC_int['Modelo'], y_train, modeloForwBIC_int['X'])
x_test_modeloForwBIC_int = crear_data_modelo(x_test, modeloForwBIC_int['Variables']['cont'], modeloForwBIC_int['Variables']['categ'], modeloForwBIC_int['Variables']['inter'])
r_test_forwBIC_int = Rsq(modeloForwBIC_int['Modelo'], y_test, x_test_modeloForwBIC_int)
param_forwBIC_int = len(modeloForwBIC_int['Modelo'].params)
```

Listing 13: Creación de modelos de regresión lineal por selección de variables clásica.

Los resultados más importantes de estos modelos se recogen en la tabla 1. De esta tabla se puede concluir que los mejores modelos son aquellos que emplean los métodos Forward y Stepwise

con métrica BIC, ya que tienen un coeficiente R^2 parecido y el menor número de parámetros. Es decir, estos dos modelos serán los mejores para predecir la variable objetivo continua.

Método	Métrica	R^2_{train}	R^2_{test}	Parámetros
Backward	AIC	0.6625	0.6364	83
Backward	BIC	0.6582	0.6335	65
Forward	AIC	0.6620	0.6377	79
Forward	BIC	0.6566	0.6343	64
Stepwise	AIC	0.6620	0.6377	79
Stepwise	BIC	0.6566	0.6343	64

Tabla 1: Resultados de R^2 en entrenamiento y test para distintos métodos y métricas

El código siguiente ejecuta un proceso de validación cruzada que permite comparar los modelos:

```
 results = pd.DataFrame({
     'Rsquared': []
     , 'Resample': []
     , 'Modelo': []})
 for rep in range(20):
     # Realiza validaci n cruzada en cuatro modelos diferentes y almacena sus
R-squared en listas separadas

     modelo_stepAIC_int = validacion_cruzada_lm(
         5
         , x_train
         , y_train
         , modeloStepAIC_int['Variables']['cont']
         , modeloStepAIC_int['Variables']['categ']
         , modeloStepAIC_int['Variables']['inter']
     )
     modelo_stepBIC_int = validacion_cruzada_lm(
         5
         , x_train
         , y_train
         , modeloStepBIC_int['Variables']['cont']
         , modeloStepBIC_int['Variables']['categ']
         , modeloStepBIC_int['Variables']['inter']
     )
     modelo_backAIC_int = validacion_cruzada_lm(
         5
         , x_train
         , y_train
         , modeloBackAIC_int['Variables']['cont']
         , modeloBackAIC_int['Variables']['categ']
         , modeloBackAIC_int['Variables']['inter']
     )
     modelo_backBIC_int = validacion_cruzada_lm(
         5
         , x_train
         , y_train
         , modeloBackBIC_int['Variables']['cont']
         , modeloBackBIC_int['Variables']['categ']
         , modeloBackBIC_int['Variables']['inter']
     )
     modelo_forwAIC_int = validacion_cruzada_lm(
         5
         , x_train
         , y_train
```

```
            , modeloForwAIC_int['Variables']['cont']
            , modeloForwAIC_int['Variables']['categ']
            , modeloForwAIC_int['Variables']['inter']
        )
        modelo_forwBIC_intt = validacion_cruzada_lm(
            5
            , x_train
            , y_train
            , modeloForwBIC_int['Variables']['cont']
            , modeloForwBIC_int['Variables']['categ']
            , modeloForwBIC_int['Variables']['inter']
        )

        results_rep = pd.DataFrame({
            'Rsquared': modelo_stepAIC_int + modelo_stepBIC_int +
    modelo_backAIC_int + modelo_backBIC_int + modelo_forwAIC_int +
    modelo_forwBIC_intt
            , 'Resample': ['Rep' + str((rep + 1))]*5*6
            , 'Modelo': [1]*5 + [2]*5 + [3]*5 + [4]*5 + [5]*5 + [6]*5
        })
        results = pd.concat([results, results_rep], axis = 0)
```

Listing 14: Modelos de regresión lineal. Validación cruzada.

Y los resultados son:

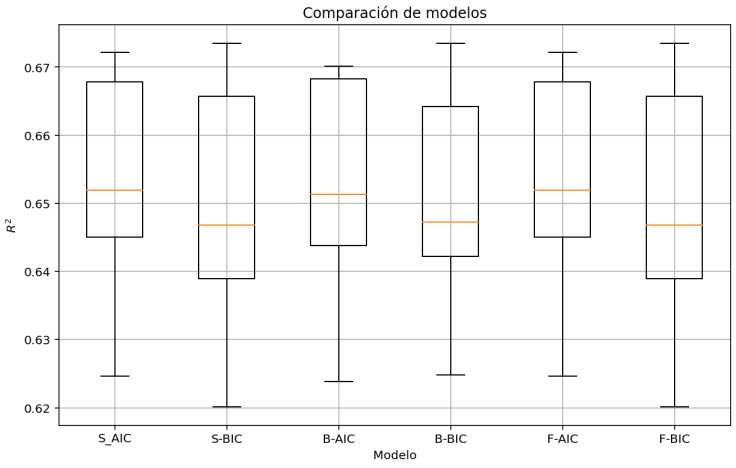

Figura 13: Diagrama de cajas de los coeficientes R^2 de los modelos seleccionados.

Modelo	$\overline{R^2}$	$\sigma(R^2)$	Parámetros
S-AIC	0.6523	0.0171	79
S-BIC	0.6490	0.0192	64
B-AIC	0.6515	0.0171	83
B-BIC	0.6504	0.0172	65
F-AIC	0.6523	0.0171	79
F-BIC	0.6490	0.0192	64

Tabla 2: Valores de promedio y desviación típica de R^2 y número de parámetros para distintos modelos

Por lo que el modelo ganador es el *Stepwise* BIC, ya que tiene el menor número de parámetros y los promedios de R^2 son similares.

4.2. Modelo de regresión lineal. Selección de variables aleatoria.

En este caso se va a ejecutar un bucle de 30 iteraciones. En cada una se hace una partición aleatoria de los datos de entrenamiento **x_train** (de los cuales se destina el 70 % para el nuevo entrenammiento y 30 % para las pruebas). Una vez divididos se desarrolla un modelo con método de selección paso a paso (*Stepwise*) y según el criterio bayesiano (BIC). Es decir, en cada iteración se genera un modelo nuevo, y del conjunto de modelos se obtienen todas las combinaciones de variables que potencialmente puedan mejorar las predicciones. En esta selección se tienen en cuenta las interacciones anteriores.

```
variables_seleccionadas = {
    'Formula': [],
    'Variables': []}
for x in range(30):
    print('----------------------------- iter: ' + str(x))
    x_train2, x_test2, y_train2, y_test2 = train_test_split(x_train, y_train,
test_size = 0.3, random_state = 1234567 + x)
    modelo = lm_stepwise(y_train2.astype(int), x_train2, var_cont, var_categ,
interacciones_unicas, 'BIC')
    variables_seleccionadas['Variables'].append(modelo['Variables'])
    variables_seleccionadas['Formula'].append(sorted(modelo['Modelo'].model.
exog_names))
variables_seleccionadas['Formula'] = list(map(lambda x: '+'.join(x),
variables_seleccionadas['Formula']))
frecuencias = Counter(variables_seleccionadas['Formula'])
frec_ordenada = pd.DataFrame(list(frecuencias.items()), columns = ['Formula',
'Frecuencia'])
frec_ordenada = frec_ordenada.sort_values('Frecuencia', ascending = False).
reset_index()
var_1 = variables_seleccionadas['Variables'][variables_seleccionadas['Formula'
].index(frec_ordenada['Formula'][0])]
```

Listing 15: Selección de variables aleatoria.

Gracias a este proceso, se pueden emplear las tres fórmulas que más veces han aparecido de forma aleatoria para el desarrollo de otros 3 modelos, que se comparan a continuación con el modelo ganador de selección clásica (S-BIC):

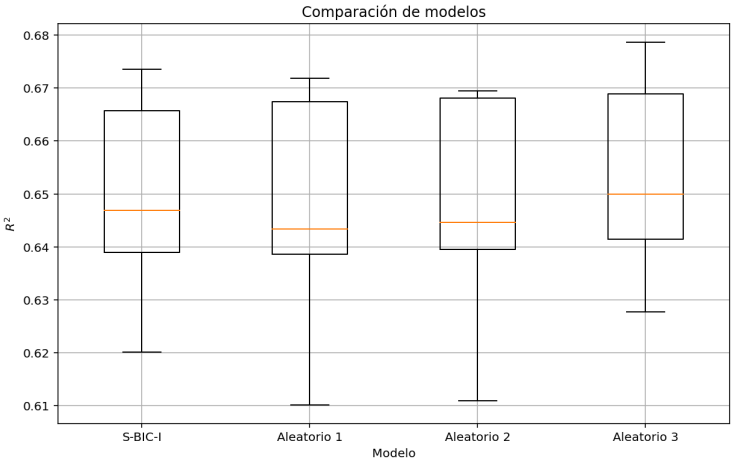

Figura 14: Comparación entre el modelo ganador de selección clásica y tres modelos aleatorios.

Y los resultados son:

Modelo	$\overline{R^2}$	$\sigma(R^2)$	Parámetros
S-BIC	0.6490	0.0192	64
Aleatorio 1	0.6462	0.0224	33
Aleatorio 2	0.6465	0.0216	35
Aleatorio 3	0.6533	0.0185	62

Tabla 3: Resultados de R^2 promedio y desviación estándar para distintos modelos.

Donde es evidente que el **primer modelo aleatorio** es el ganador por el principio de Parsimonia, ya que necesita muchos menos parámetros (aproximadamente la mitad) que los modelos S-BIC, aleatorio 2 y aleatorio 3, aunque el modelo aleatorio 2 es un buen candidato.

4.3. Modelo de regresión lineal. Conclusiones.

El resumen del modelo ganador es el siguiente. Se puede observar que el modelo ganador tiene un coeficiente F de 374,7. Se puede observar que, mientras que un aumento en la mayoría de variables supone una predicción menor para la variable **Izquierda**, hay algunas que indican aumento, como **AgricultureUnemploymentPtge** o **UnemployLess25_Ptge**. Con respecto al valor del estadístico p, la mayoría de variables son relevantes al modelo pues este sólo supera el 5 % en dos ocasiones (ambas para factores relativos a interacciones entre variables).

```
==============================================================================
Dep. Variable:              Izda_Pct   R-squared:                       0.650
Model:                           OLS   Adj. R-squared:                  0.648
Method:                Least Squares   F-statistic:                     374.7
Date:               Thu, 20 Feb 2025   Prob (F-statistic):               0.00
Time:                       22:16:44   Log-Likelihood:                -23995.
No. Observations:               6493   AIC:                         4.806e+04
Df Residuals:                   6460   BIC:                         4.828e+04
Df Model:                         32
Covariance Type:            nonrobust
==============================================================================================================
                                                 coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------------------------------
const                                         34.0268      2.446     13.909      0.000      29.231      38.822
raiz4DifComAutonPtge                          10.3711      0.721     14.391      0.000       8.958      11.784
sqrtxAgricultureUnemploymentPtge               1.9167      0.254      7.537      0.000       1.418       2.415
Age_19_65_pct                                  0.1849      0.022      8.430      0.000       0.142       0.228
sqrtxUnemployLess25_Ptge                       6.0633      0.720      8.420      0.000       4.652       7.475
logxPopulation                                 0.6471      0.129      5.013      0.000       0.394       0.900
raiz4Explotaciones                            -5.5464      0.746     -7.432      0.000      -7.009      -4.083
ForeignersPtge                                -0.1147      0.019     -6.066      0.000      -0.152      -0.078
logxinmuebles                                  0.8258      0.154      5.356      0.000       0.524       1.128
logxConstructionUnemploymentPtge               0.1921      0.038      5.035      0.000       0.117       0.267
sqrtxWomanPopulationPtge                       2.2544      0.574      3.925      0.000       1.129       3.380
CCAA_Aragón                                   -9.3148      0.659    -14.133      0.000     -10.607      -8.023
CCAA_Asturias                                 -6.0731      1.325     -4.582      0.000      -8.671      -3.475
CCAA_Baleares                                 -8.7071      1.439     -6.051      0.000     -11.528      -5.886
CCAA_Canarias                                -12.7730      1.299     -9.836      0.000     -15.319     -10.227
CCAA_Cantabria                               -16.7808      1.225    -13.700      0.000     -19.182     -14.380
CCAA_CastillaLeón                            -18.7859      0.562    -33.422      0.000     -19.888     -17.684
CCAA_CastillaMancha                          -11.5004      0.629    -18.273      0.000     -12.734     -10.267
CCAA_Cataluña                                -42.8001      0.622    -68.781      0.000     -44.020     -41.580
CCAA_ComValenciana                           -30.2972      0.668    -45.328      0.000     -31.608     -28.987
CCAA_Extremadura                              -4.7275      0.716     -6.603      0.000      -6.131      -3.324
CCAA_Galicia                                 -30.4978      0.827    -36.889      0.000     -32.118     -28.877
CCAA_Madrid                                  -15.6571      0.987    -15.857      0.000     -17.593     -13.721
CCAA_Murcia                                  -15.5734      1.613     -9.654      0.000     -18.736     -12.411
CCAA_Navarra                                 -12.9485      0.839    -15.433      0.000     -14.593     -11.304
CCAA_PaísVasco                               -23.1846      0.845    -27.422      0.000     -24.842     -21.527
CCAA_Rioja                                   -14.2929      0.999    -14.312      0.000     -16.251     -12.335
sqrtxUnemployLess25_Ptge_ActividadPpal_Otro     4.3986      0.804      5.471      0.000       2.823       5.975
sqrtxUnemployLess25_Ptge_ActividadPpal_Servicios_Construccion_Industria    -0.6720      2.652     -0.253      0.800      -5.870       4.527
UnemployLess25_Ptge_ActividadPpal_Otro        -0.6543      0.070     -9.292      0.000      -0.792      -0.516
UnemployLess25_Ptge_ActividadPpal_Servicios_Construccion_Industria    -0.2955      0.332     -0.891      0.373      -0.945       0.354
sqrtxUnemployLess25_Ptge_Age_under19_Ptge     -0.6679      0.066    -10.087      0.000      -0.798      -0.538
UnemployLess25_Ptge_Age_under19_Ptge           0.0449      0.005      8.836      0.000       0.035       0.055
==============================================================================
Omnibus:                      394.304   Durbin-Watson:                   2.013
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              825.039
Skew:                           0.413   Prob(JB):                    7.00e-180
Kurtosis:                       4.539   Cond. No.                     3.69e+03
==============================================================================
```

Figura 15: Modelo de regresión lineal ganador.

5. Modelos de regresión logística.

En esta sección se repite todo el proceso del apartado 4. Para ello se emplean funciones de **FuncionesMineria** parecidas pero adaptadas a la variable objetivo binaria. El proceso es el mismo: selección de variables clásica, selección aleatoria y comparación. Mediante validación cruzada se elegirán los mejores modelos para predecir la variable **Izquierda**.

```
with open('datosTarea_bin.pickle', 'rb') as f:
    todo = pickle.load(f)
varObjBin = todo['Izquierda']
todo = todo.drop('Izquierda', axis = 1)
pd.DataFrame({
    'n': varObjBin.value_counts(),
    '%': varObjBin.value_counts(normalize = True)})
var_cont = todo.select_dtypes(include = ['int', 'float']).columns.tolist()
var_cont_sin_transf = ['Population','TotalCensus', ..., 'Explotaciones']
var_cont_transf = [var for var in var_cont if var not in var_cont_sin_transf]
var_categ = [var for var in todo.columns.tolist() if var not in var_cont]
todo = todo.drop(var_cont_transf, axis = 1)
# Partici n de datos. La variable respuesta se pone como num rica:
x_train, x_test, y_train, y_test = train_test_split(todo, varObjBin, test_size
 = 0.2, random_state = 1234567)
y_train, y_test = y_train.astype(int), y_test.astype(int)
```

Listing 16: Modelo de regresión logística. Asignación de varibales y partición de datos.

5.1. Modelo de regresión logística. Selección de variables clásica.

Para la obtención de modelos por selección clásica se ejecuta el código siguiente. En este caso no se contemplan modelos de selección hacia adelante (*Forward*) porque, tal y como se ha visto en el apartado 4.1 proporcionan los mismos resultados que los modelos con selección paso a paso. El código para crear estos modelos es el siguiente, donde no se han tenido en cuenta interacciones debido a problemas de capacidad de computación:

```
interacciones_unicas = []

# MODELO STEPWISE AIC.
modeloStepAIC = glm_stepwise(y_train, x_train, var_cont_sin_transf, var_categ,
 [], 'AIC')
summary_glm(modeloStepAIC['Modelo'], y_train, modeloStepAIC['X'])
r_train_stepAIC = pseudoR2(modeloStepAIC['Modelo'], modeloStepAIC['X'],
y_train)
x_test_modeloStepAIC = crear_data_modelo(x_test, modeloStepAIC['Variables']['
cont'], modeloStepAIC['Variables']['categ'], modeloStepAIC['Variables']['inter'
])
r_test_stepAIC = pseudoR2(modeloStepAIC['Modelo'], x_test_modeloStepAIC,
y_test)
param_stepAIC = len(modeloStepAIC['Modelo'].coef_[0])

# MODELO STEPWISE BIC.
modeloStepBIC = glm_stepwise(y_train, x_train, var_cont_sin_transf, var_categ,
 [], 'BIC')
summary_glm(modeloStepBIC['Modelo'], y_train, modeloStepBIC['X'])
r_train_stepBIC = pseudoR2(modeloStepBIC['Modelo'], modeloStepBIC['X'],
y_train)
x_test_modeloStepBIC = crear_data_modelo(x_test, modeloStepBIC['Variables']['
cont'], modeloStepBIC['Variables']['categ'], modeloStepBIC['Variables']['inter'
])
```

```
r_test_stepBIC = pseudoR2(modeloStepBIC['Modelo'], x_test_modeloStepBIC, y_test)
param_stepBIC = len(modeloStepBIC['Modelo'].coef_[0])

# MODELO BACKWARD AIC.
modeloBackAIC = glm_backward(y_train, x_train, var_cont_sin_transf, var_categ, [], 'AIC')
summary_glm(modeloBackAIC['Modelo'], y_train, modeloBackAIC['X'])
r_train_backAIC = pseudoR2(modeloBackAIC['Modelo'], modeloBackAIC['X'], y_train)
x_test_modeloBackAIC = crear_data_modelo(x_test, modeloBackAIC['Variables']['cont'], modeloBackAIC['Variables']['categ'], modeloBackAIC['Variables']['inter'])
r_test_backAIC = pseudoR2(modeloBackAIC['Modelo'], x_test_modeloBackAIC, y_test)
param_backAIC = len(modeloBackAIC['Modelo'].coef_[0])

# MODELO BACKWARD BIC.
modeloBackBIC = glm_backward(y_train, x_train, var_cont_sin_transf, var_categ, [], 'BIC')
summary_glm(modeloBackBIC['Modelo'], y_train, modeloBackBIC['X'])
r_train_backBIC = pseudoR2(modeloBackBIC['Modelo'], modeloBackBIC['X'], y_train)
x_test_modeloBackBIC = crear_data_modelo(x_test, modeloBackBIC['Variables']['cont'], modeloBackBIC['Variables']['categ'], modeloBackBIC['Variables']['inter'])
r_test_backBIC = pseudoR2(modeloBackBIC['Modelo'], x_test_modeloBackBIC, y_test)
param_backBIC = len(modeloBackBIC['Modelo'].coef_[0])
```

Listing 17: Modelo de regresión logística. Creación de modelos por selección de variables clásica.

De nuevo, se genera una tabla con los resultados de cada modelo en función del proceso y la métrica empleados. :

Método	Métrica	R^2_{train}	R^2_{test}	Parámetros
Backward	AIC	0.6564	0.6126	73
Backward	BIC	0.6547	0.6203	65
Stepwise	AIC	0.6549	0.6340	68
Stepwise	BIC	0.6538	0.6336	63

Tabla 4: R^2 en modelos básicos de regresión logística.

A continuación se realiza un proceso de validación cruzada para comprobar qué modelo es mejor, obteniendo así una tabla con los promedios y las desviaciones típicas de los coeficientes R^2.

```
results = pd.DataFrame({
    'AUC': [],
    'Resample': [],
    'Modelo': []})

for rep in range(20):
    modelo_StepAIC = validacion_cruzada_glm(5, x_train, y_train, modeloStepAIC['Variables']['cont'], modeloStepAIC['Variables']['categ'], modeloStepAIC['Variables']['inter'])
    modelo_StepBIC = validacion_cruzada_glm(5, x_train, y_train, modeloStepBIC['Variables']['cont'], modeloStepBIC['Variables']['categ'],  modeloStepBIC['Variables']['inter'])
    modelo_BackAIC = validacion_cruzada_glm(5, x_train, y_train, modeloBackAIC['Variables']['cont'], modeloBackAIC['Variables']['categ'], modeloBackAIC['Variables']['inter'])
```

```
        modelo_BackBIC= validacion_cruzada_glm(5, x_train, y_train, modeloBackBIC['Variables']['cont'], modeloBackBIC['Variables']['categ'], modeloBackBIC['Variables']['inter'])

        results_rep = pd.DataFrame({
            'AUC': modelo_StepAIC + modelo_StepBIC + modelo_BackAIC + modelo_BackBIC
            ,'Resample': ['Rep' + str((rep + 1))]*5*4
            , 'Modelo': [1]*5 + [2]*5 + [3]*5 + [4]*5})
        results = pd.concat([results, results_rep], axis = 0)
```

Listing 18: Modelo de regresión logística. Validación cruzada por selección de variables clásica.

Y los resultados obtenidos son los siguientes:

Modelo	$\overline{R^2}$	$\sigma(R^2)$	Parámetros
S-AIC	0.8625	0.0116	76
S-BIC	0.8625	0.0116	76
B-AIC	0.8642	0.0115	75
B-BIC	0.8642	0.0115	75

Tabla 5: Valores de promedio y desviación típica de R^2 y número de parámetros para distintos modelos

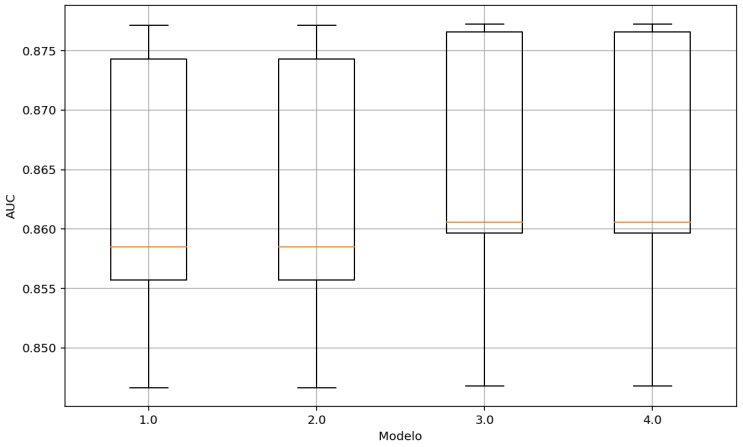

Figura 16: Gráfico de cajas para los modelos de regresión logística creados por selección de variables clásica.

Por lo que se puede afirmar que el modelo ganador es aquel que se ha desarrollado con el método *Backward* y la métrica BIC. Este modelo se comparará con tres obtenidos mediante selección aleatoria de variables en el siguiente apartado.

5.2. Modelo de regresión logística. Selección de variables aleatoria.

El código para la selección aleatoria es el siguiente:

```
    variables_seleccionadas = {
        'Formula': [],
        'Variables': [],
        'Interaccion': []}

    for x in range(20):
        print('---------------------------- iter: ' + str(x))
        x_train2, x_test2, y_train2, y_test2 = train_test_split(x_train, y_train, test_size = 0.3, random_state = 1234567 + x)
```

```
        modelo = glm_stepwise(y_train2.astype(int), x_train2, var_cont, var_categ, interacciones_unicas, 'BIC')
        variables_seleccionadas['Variables'].append(modelo['Variables'])
        variables_seleccionadas['Formula'].append(sorted(modelo['X'].columns))
    variables_seleccionadas['Formula'] = list(map(lambda x: '+'.join(x), variables_seleccionadas['Formula']))

    frecuencias = Counter(variables_seleccionadas['Formula'])
    frec_ordenada = pd.DataFrame(list(frecuencias.items()), columns = ['Formula', 'Frecuencia'])
    frec_ordenada = frec_ordenada.sort_values('Frecuencia', ascending = False).reset_index()

    var_1 = variables_seleccionadas['Variables'][variables_seleccionadas['Formula'].index(frec_ordenada['Formula'][0])]
    var_2 = variables_seleccionadas['Variables'][variables_seleccionadas['Formula'].index(frec_ordenada['Formula'][1])]
    var_3 = variables_seleccionadas['Variables'][variables_seleccionadas['Formula'].index(frec_ordenada['Formula'][2])]
```

Listing 19: Modelo de regresión logística. Selección de variables aleatoria.

De nuevo se ejecuta una validación cruzada del modelo ganador por selección clásica y los modelos aleatorios más fiables, obteniendo el siguiente gráfico de cajas y la tabla de coeficientes estadísticos:

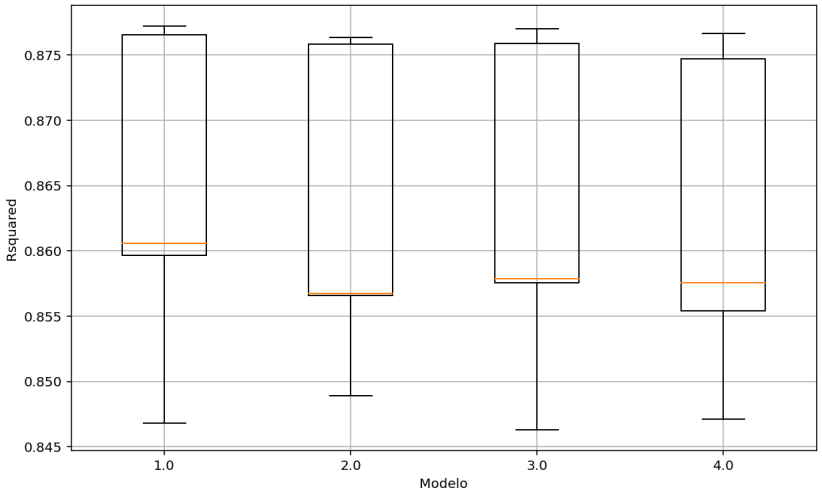

Figura 17: Validación cruzada para modelos de regresión logística aleatorios.

Modelo	$\overline{R^2}$	$\sigma(R^2)$	Parámetros
B-BIC	0.8625	0.0116	75
Aleatorio 1	0.8625	0.0116	74
Aleatorio 1	0.8642	0.0115	75
Aleatorio 3	0.8642	0.0115	74

Tabla 6: Valores de promedio y desviación típica de R^2 y número de parámetros para distintos modelos

Por lo que se elige como modelo ganador al **aleatorio 3**.

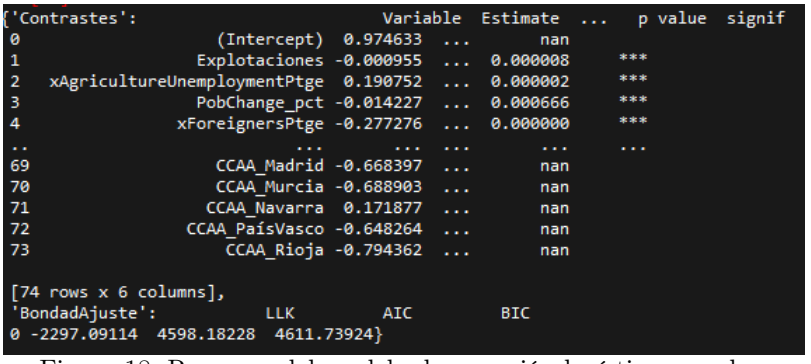

Figura 18: Resumen del modelo de regresión logística ganador.

5.3. Modelo de regresión logística. Conclusiones.

El resumen del modelo ganador es el siguiente, donde se puede ver el efecto de todas las variables independientes sobre la variable objetivo binaria **Izquierda**.

6. Anexo.

En esta sección se incluyen tablas, gráficos o celdas de código adicionales que pueden servir para un mejor entendimiento de la práctica.

Variable	Izda_Pct	Izquierda
CodigoProvincia	0.1253	0.1609
CCAA	0.4876	0.5614
Population	0.0897	0.1322
TotalCensus	0.0946	0.1426
Age_0-4_Ptge	0.1107	0.1308
Age_under19_Ptge	0.1101	0.1401
Age_19$_6$5_*pct*	0.0855	0.1227
Age_over65_pct	0.1086	0.1435
WomanPopulationPtge	0.0840	0.1227
ForeignersPtge	0.0737	0.1017
SameComAutonPtge	0.0886	0.0845
SameComAutonDiffProvPtge	0.1280	0.1060
DifComAutonPtge	0.0918	0.0308
UnemployLess25_Ptge	0.2037	0.2688
Unemploy25_40_Ptge	0.0883	0.1282
UnemployMore40_Ptge	0.1238	0.1855
AgricultureUnemploymentPtge	0.1448	0.1998
IndustryUnemploymentPtge	0.1118	0.1141
ConstructionUnemploymentPtge	0.0676	0.0649
ServicesUnemploymentPtge	0.0633	0.0842
totalEmpresas	0.0910	0.1158
Industria	0.0679	0.0134
Construccion	0.0829	0.0250
ComercTTEHosteleria	0.0625	0.0175
Servicios	0.0716	0.0085
ActividadPpal	0.1200	0.1759
inmuebles	0.1047	0.1471
Pob2010	0.0948	0.1346
SUPERFICIE	0.1178	0.1604
Densidad	0.0666	0.0320
PobChange_pct	0.0777	0.1149
PersonasInmueble	0.0773	0.0927
Explotaciones	0.1165	0.1761
prop_missings	0.0552	0.0608

Tabla 7: Coeficientes V de Cramer para las variables objetivo.