7-2-2025

MINERÍA DE DATOS Y MODELIZACIÓN PREDICTIVA

Depuración de datos, modelos de regresión lineal y logística

Jacobo Álvarez Gutiérrez
MÁSTER BIG DATA & IA (UCM)

Contenido

1 Introducción

El presente trabajo tiene como objetivo analizar y modelar datos relacionados con las elecciones municipales en España, utilizando técnicas de regresión lineal y logística para explorar factores que influyen en la abstención electoral. Para ello, se seleccionaron dos variables objetivo específicas del conjunto de datos proporcionado:

- **AbstentionPtge:** una variable continua que mide el porcentaje de abstención electoral en cada municipio.
- **AbstencionAlta:** una variable dicotómica que toma el valor 1 si el porcentaje de abstención supera el 30% y 0 en caso contrario.

El propósito del análisis es identificar y comprender las principales relaciones y patrones que explican la abstención electoral en los municipios, utilizando modelos predictivos para evaluar tanto la magnitud de la abstención como las probabilidades de que esta sea alta. Esto permitirá ofrecer una visión cuantitativa y cualitativa de los factores asociados al fenómeno, contribuyendo a la comprensión de un aspecto clave de la participación ciudadana.

El conjunto de datos utilizado combina información demográfica y socioeconómica de los distintos municipios de España. Las variables implicadas incluyen datos tanto continuos como categóricos, proporcionando un marco diverso para explorar correlaciones entre las características de los municipios y los resultados de abstención.

2 Depuración de Datos

En este apartado se detallan los procedimientos aplicados para depurar el conjunto de datos, corrigiendo errores e imperfecciones antes de la construcción de los modelos predictivos.

2.1 Importación del conjunto de datos y asignación correcta de los tipos de variables

Importamos los datos con el comando:

```
datos = pd.read_excel('DatosEleccionesEspaña.xlsx')
```

En primer lugar, eliminamos las variables que también eran candidatas a ser objetivo de estudio y no usaremos como variables explicativas:

```
var_obj_descartadas = ['Izda_Pct', 'Dcha_Pct', 'Otros_Pct',
                       'Izquierda', 'Derecha']
for i in var_obj_descartadas:
    datos.drop(i, axis=1, inplace=True)
```

A continuación, estudiamos la tipología de cada uno de los campos presentes en la base de datos a partir del comando:

```
datos.dtypes
```

Tras la ejecución de este, comprobamos que a algunos campos se les ha asignado una tipología diferente a la esperada. Concretamente, los campos 'AbstencionAlta' y 'CodigoProvincia' se han definido, erróneamente, como variables numéricas.

```
CodigoProvincia                 int64
CCAA                           object
Population                      int64
TotalCensus                     int64
AbstentionPtge                float64
AbstencionAlta                  int64
```

Por tanto, las redefinimos como variables categóricas a partir del comando:

```
datos['AbstencionAlta'] = datos['AbstencionAlta'].astype(str)
datos['CodigoProvincia'] = datos['CodigoProvincia'].astype(str)
```

2.2 Análisis descriptivo del conjunto de datos

A continuación, se hará un análisis descriptivo de las variables numéricas y categóricas. Concretamente, usamos el comando:

```
cuentaDistintos(datos)
```

Este nos permite verificar que ninguna de las variables numéricas tiene tan pocos resultados diferentes como para plantearnos el redefinirla como variable categórica. Por otra parte, usamos el comando:

```
analizar_variables_categoricas(datos)
```

Este nos ofrece un análisis descriptivo de todas las variables que hemos asignado como categóricas.

```
'ActividadPpal':                        n         %
Otro                 4932  0.607614
ComercTTEHosteleria  2538  0.312677
Servicios             620  0.076383
Construccion           14  0.001725
Industria              13  0.001602,
'Densidad':             n         %
MuyBaja  6416  0.790440
Baja     1053  0.129728
Alta      556  0.068498
?          92  0.011334}
```

En particular, nos permite identificar valores missing (?), tal y como ocurre con la variable 'Densidad'. Asimismo, vemos que las categorías "Construcción" e "Industria" de la variable 'ActividadPpal' están poco representadas en comparación con las demás, lo cual nos plantea la posibilidad de reagrupar en otras categorías.

Por otra parte, en lo que respecta a las variables numéricas, hacemos un análisis descriptivo a partir del comando:

```
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)
```

Esto nos ofrece el siguiente resultado:

Índice	count	mean	std	min	25%	50%	75%	max
Age_19_65_pct	8117	57.3706	6.81864	23.459	53.845	58.655	61.818	100.002
Age_over65_pct	8117	29.0653	11.767	-18.052	19.827	27.559	36.911	76.472
WomanPopulationPtge	8117	47.3023	4.36235	11.765	45.725	48.485	50	72.683
ForeignersPtge	8117	5.61832	7.3487	-8.96	1.06	3.59	8.18	71.47

Nótese que hay algunos campos que representan porcentajes y, sin embargo, tienen valores fuera del intervalo cerrado [0, 100]. Por otra parte, vemos también que hay valores máximos 99999 que se corresponden con valores missing.

2.3 Corrección de los errores detectados

En lo que respecta a las variables categóricas, reemplazamos los valores missing (?) por valores missing (Nan). Además, incluiremos las categorías "Construcción" e "Industria" en la categoría "Otros" de la variable 'ActividadPpal'. Esto lo hacemos a partir de las siguientes líneas de código:

```
datos['Densidad'] = datos['Densidad'].replace('?', np.nan)
datos['ActividadPpal'] = datos['ActividadPpal'].replace({'Construccion': 'Otro',
                                                         'Industria': 'Otro'})
```

Por otra parte, en lo que respecta a las variables numéricas, sustituimos por valores missing (Nan) tanto los valores de porcentajes fuera del rango [0, 100] como los valores erróneos 99999. Esto lo haremos a partir de las siguientes líneas de código:

```
ptge_outofrange = ['Age_19_65_pct', 'Age_over65_pct', 'ForeignersPtge',
                   'SameComAutonPtge']
for i in ptge_outofrange:
    datos[i] = [x if 0 <= x <= 100 else np.nan for x in datos[i]]

datos['Explotaciones'] = datos['Explotaciones'].replace(99999, np.nan)
```

Llegados a este punto, antes de pasar al tratamiento de valores atípicos y valores missing, nos interesa declarar las variables objetivo y eliminarlas del conjunto de datos. Asimismo, usaremos los nombres de los municipios como identificadores de cada registro, pues son valores que no aportan ningún tipo de información sobre los datos. Además, por comodidad para posteriores análisis, guardamos el nombre de las distintas variables distinguiendo las categóricas y las numéricas.

```
datos = datos.set_index(datos['Name']).drop('Name', axis = 1)
varObjCont = datos['AbstentionPtge']
varObjBin = datos['AbstencionAlta']
datos_input = datos.drop(['AbstentionPtge', 'AbstencionAlta'], 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]
```

2.4 Datos atípicos

Aplicamos el siguiente comando:

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

Este nos devuelve la siguiente información:

Clave/Tecla	Tipo	Tamaño	Valor
SameComAutonDiffProvPtge	float	1	0.02032770728101515 3
SameComAutonPtge	float	1	0.0
ServicesUnemploymentPtge	float	1	0.0
Servicios	float	1	0.0
SUPERFICIE	float	1	0.0
TotalCensus	float	1	0.09634101268941728
totalEmpresas	float	1	0.0
Unemploy25_40_Ptge	float	1	0.0
UnemployLess25_Ptge	float	1	0.003203153874584206

Tal y como se puede apreciar, la proporción de valores atípicos para cada una de las variables es pequeña. Por ende, podemos considerarlos valores atípicos y la forma de proceder será considerarlos como valores missing. Haremos dicha transformación a partir de la siguiente línea de código:

```
for x in numericas_input:
    datos_input[x] = atipicosAmissing(datos_input[x])[0]
```

2.5 Datos Missing

En primer lugar, calculamos la proporción de valores missing por cada variable a partir del siguiente comando:

```
prop_missingsVars = datos_input.isna().sum()/len(datos_input)
```

Esto nos devuelve la siguiente información:

Índice	0
CodigoProvincia	0
CCAA	0
Population	0.0990514
TotalCensus	0.096341
Age_0-4_Ptge	0
Age_under19_Ptge	0

Tras analizar todas las variables, se aprecia que en ninguno de los casos hay más de un 10% de los valores perdidos, así que descartamos por completo la eliminación de dichas variables.

Por otra parte, analizamos también la proporción de valores perdidos para cada registro de la base de datos. Esto se hace a partir del siguiente comando:

```
datos_input['prop_missings'] = datos_input.isna().mean(axis = 1)
datos_input['prop_missings'].describe()
```

El resultado obtenido es el siguiente:

```
count    8117.000000
mean        0.013716
std         0.025246
min         0.000000
25%         0.000000
50%         0.000000
75%         0.030303
max         0.333333
```

Nótese que los registros que más valores perdidos tienen presentan el 66,66% de los datos, de modo que descartamos también la opción de eliminar alguno de ellos. Por otra parte, del análisis de esta nueva variable 'prop_missings', vemos que existen solo 9 valores diferentes. Esto da pie a incluirla en el conjunto de datos como variable categórica. No obstante, alguna de estas categorías no está suficientemente representada, así que la reorganizamos como una variable dicotómica: '0' (No existen datos missing) y '1' (Existen datos missing) de forma análoga a como se hizo con la variable 'ActividadPpal'.

Tras todo el análisis previo, concluimos con la imputación de valores para estos valores missing. Aunque este proceso se podría hacer con un poco más de detalle, por simplicidad, se hará la imputación de manera aleatoria tanto para las variables categóricas como numéricas. Se usa para ello el siguiente código:

```
for x in categoricas_input:
    datos_input[x] = ImputacionCuali(datos_input[x], 'aleatorio')

for x in numericas_input:
    datos_input[x] = ImputacionCuant(datos_input[x], 'aleatorio')
```

En este punto, usamos el siguiente comando para comprobar que no queda ningún valor missing:

```
datos_input.isna().sum()
```

En efecto, el resultado obtenido es el siguiente:

```
CodigoProvincia          0
CCAA                     0
Population               0
TotalCensus              0
Age_0-4_Ptge             0
Age_under19_Ptge         0
Age_19_65_pct            0
```

Finalmente, podemos afirmar que hemos concluido con la depuración de los datos. Solo nos resta guardar estos datos depurados en otro archivo diferente para no perder los datos originales en caso de querer recuperarlos para, por ejemplo, realizar una imputación de valores missing con mayor detalle. Esto se hará a partir de las siguientes líneas de código:

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

3 Detección de relaciones entre variables

La finalidad de este apartado es hacer una selección previa de las variables más representativas de cara a la modelización de las variables objetivo. Para ello, usaremos tanto el coeficiente de correlación de Pearson como el estadístico V de Cramer.

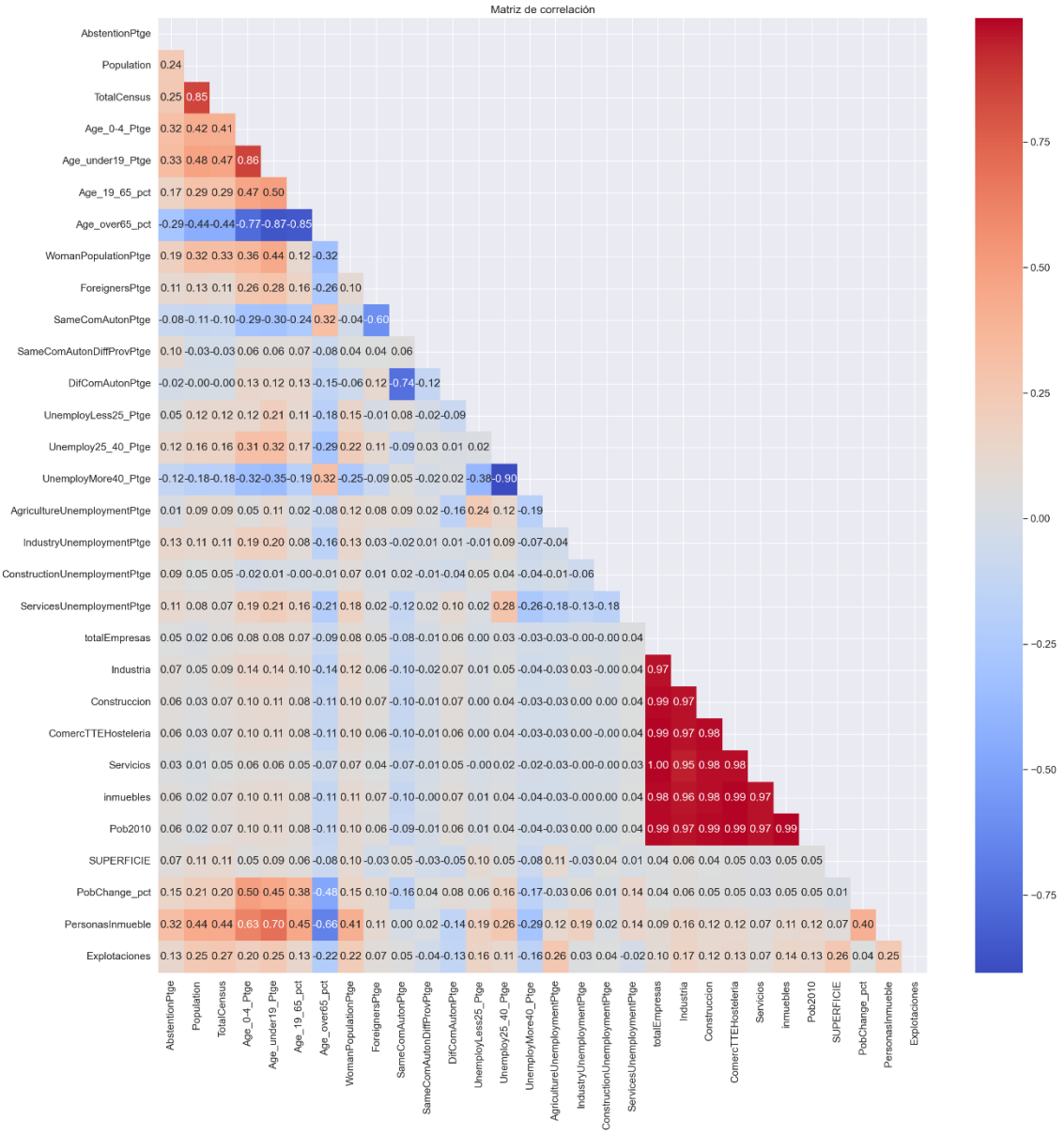
En primer lugar, hacemos una llamada a los datos depurados obtenidos en el apartado anterior y separamos de nuevo los valores de las variables objetivos del resto del conjunto de datos:

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

varObjCont = datos['AbstentionPtge']
varObjBin = datos['AbstencionAlta']
datos_input = datos.drop(['AbstentionPtge', 'AbstencionAlta'], axis = 1)
```

Empezaremos con el coeficiente de correlación de Pearson. Para ello tenemos las líneas de código que siguen a continuación:

```
numericas = datos_input.select_dtypes(include=['int', 'float']).columns
matriz_corr = pd.concat([varObjCont, datos_input[numericas]],
                        axis = 1).corr(method = 'pearson')
mask = np.triu(np.ones_like(matriz_corr, dtype=bool))
plt.figure(figsize=(22, 22))
sns.set(font_scale=1.2)
sns.heatmap(matriz_corr, annot=True, cmap='coolwarm', fmt=".2f",
            cbar=True, mask=mask)
plt.title("Matriz de correlación")
plt.show()
```

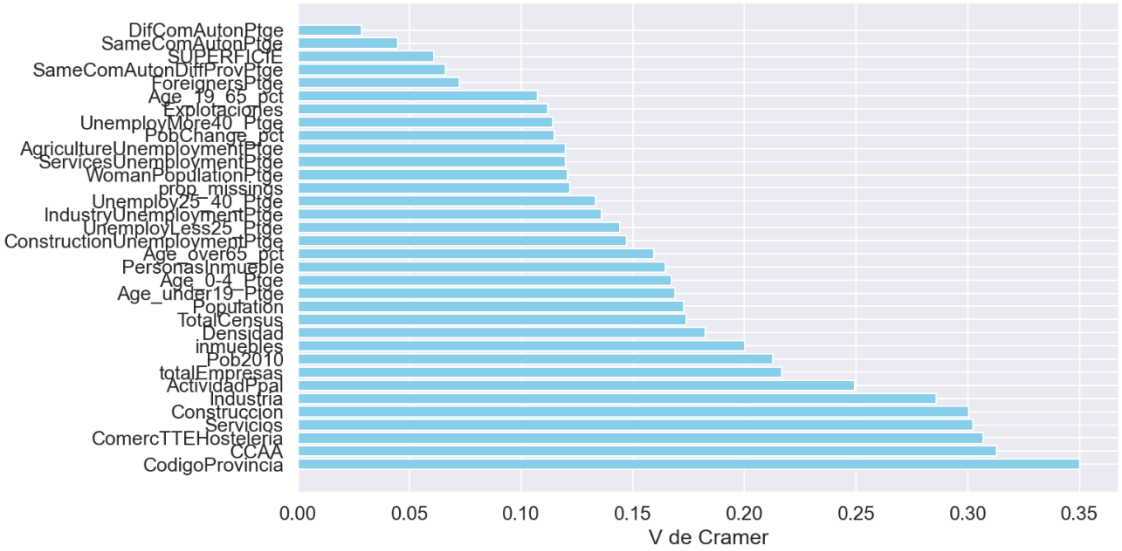
Esto nos ofrece la siguiente representación gráfica:

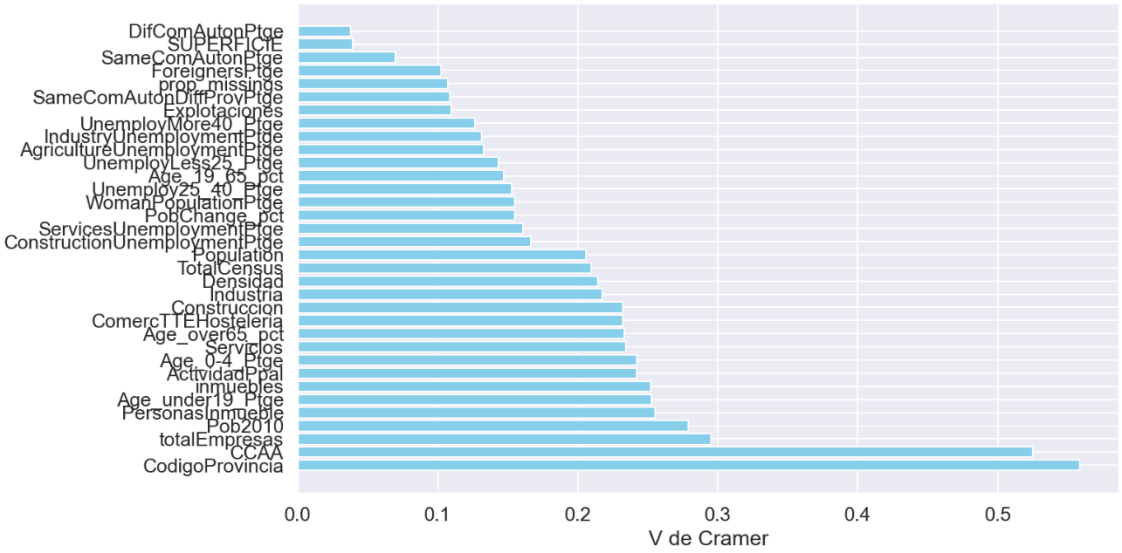


Veamos a continuación la información que refleja el estadístico V de Cramer a partir del siguiente código:

```
graficoVcramer(datos_input, varObjCont)
graficoVcramer(datos_input, varObjBin)
```

El resultado obtenido para la variable objetivo numérica y la categórica es, respectivamente:





Como se puede apreciar de los valores del coeficiente de correlación de Pearson, muchas de las variables no demuestran una correlación lineal con la variable objetivo 'AbstentionPtge'. De hecho, estableceremos el criterio de quedarnos solo con aquellas cuyo coeficiente (en valor absoluto) supere el umbral 0.25, lo cual nos limita a las variables: 'TotalCensus', 'Age_0-4_Ptge', 'Age_under19_Ptge', 'Age_over65_pct' y 'PersonasInmueble'. Vemos también en ese primer gráfico que existe una aparente redundancia entre la variable 'Population' y 'TotalCensus', así que descartaremos la primera por tener una menor correlación con la variable objetivo (a pesar de ser un valor muy cercano al umbral establecido). Además, también se aprecia una redundancia entre aquellas variables que representan las diferentes actividades socioeconómicas, de modo que, en consonancia con los resultados del estadístico V de Cramer, resumiremos la influencia de estas quedándonos con las variables 'totalEmpresas' y 'ActividadPpal'. Finalmente, de la información que nos ofrecen estos dos últimos gráficos, consideraremos también las variables 'CCAA' y 'CodigoProvincia'.

Resumidamente, de cara a empezar con la modelización predictiva de ambas variables objetivo, trabajaremos en primera instancia con la siguiente colección de variables:

```
var_cont = ['totalEmpresas', 'TotalCensus', 'PersonasInmueble',
            'Age_0-4_Ptge', 'Age_under19_Ptge', 'Age_over65_pct']
var_categ = ['CCAA', 'CodigoProvincia', 'ActividadPpal']
```

4 Modelo de regresión lineal

Lo primero que haremos para el desarrollo de un modelo de regresión lineal será una partición de los datos. Concretamente, trabajaremos con el 80% de estos para entrenar el modelo y reservaremos el 20% restante a modo de test. Esto lo haremos a partir de las siguientes líneas de código:

```
x_train, x_test, y_train, y_test = train_test_split(datos_input,
                                                    np.ravel(varObjCont),
                                                    test_size = 0.2,
                                                    random_state = 123456)
```

A continuación, se irán aplicando sucesivamente cada uno de los métodos de selección de variables (stepwise, backward y forward). Hay que mencionar que, dada la extensión del resumen de cada modelo que ofrece la consola, solo se registrarán algunos valores relevantes para el estudio. Concretamente, se registrarán el valor del estadístico F y su p-valor, los valores del coeficiente de correlación R^2 tanto para los datos de entrenamiento como los datos test, el estadístico Durbin-Watson y el estadístico Jarque-Bera con su p-valor asociado. Además, de nuevo por simplicidad y, atendiendo a que no existe una diferencia significativa de estos valores entre los diferentes modelos generados, solo se registrarán los mismos para el primero de los modelos. Esto será suficiente para explicar la relevancia y el significado de cada uno de estos parámetros. Una vez se seleccione el modelo ganador, sí que se mostrará en detalle todo el conjunto de parámetros de este.

4.1 Métodos de selección clásica de variables

A continuación, se muestran las líneas de código para implementar uno de los métodos de selección clásico de variables. Solo se hará con uno de ellos por simplificar el documento de texto (la implementación de los demás es análoga, pero usando las funciones propias *lm_backward* y *lm_forward*, así como modificando las métricas AIC o BIC).

```
# ModeloStepwise(AIC)

modeloStepAIC = lm_stepwise(y_train, x_train, var_cont, var_categ, [], 'AIC')
modeloStepAIC['Modelo'].summary()
Rsq(modeloStepAIC['Modelo'], y_train, modeloStepAIC['X'])
x_test_modeloStepAIC = crear_data_modelo(x_test, modeloStepAIC['Variables']['cont'],
                                                 modeloStepAIC['Variables']['categ'],
                                                 modeloStepAIC['Variables']['inter'])
Rsq(modeloStepAIC['Modelo'], y_test, x_test_modeloStepAIC)
```

Los valores más relevantes comentados en el párrafo introductorio de esta sección resultan ser:

F-statistic	**Prob(F)**	**R^2 train**	**R^2 test**
78.07	0.00	0.404	0.399
Durbin-Watson		**Jarque-Bera**	**Prob(J-B)**
2.028		610.980	$2.12e-133$

Aunque no se muestran los valores de los parámetros para todos los modelos generados, es relevante comentar que todos los modelos tienen un p-valor asociado al estadístico F nulo, lo cual indica que al menos una de las variables tiene relación directa con la variable objetivo. Por otra parte, en lo que respecta a los residuos del modelo, vemos que el estadístico de Durbin-Watson es próximo a 2 en todos los casos, lo cual es un indicativo de la no correlación de los

residuos. Además, vemos que el p-valor asociado al estadístico de Jarque-Bera es ínfimo, lo cual denota la no normalidad en la distribución de estos. Por otra parte, vemos que los valores del coeficiente de correlación R^2 es muy próximo tanto para el conjunto de datos de entrenamiento como test en todos los modelos, lo cual es un buen indicativo de la robustez de estos.

A continuación, probaremos a generar interacciones entre las variables numéricas añadiendo la siguiente variable:

```
interacciones = list(itertools.combinations(var_cont, 2))
```

Ahora, la aplicación del modelo sería similar a la entrada de código que se mostró anteriormente, pero sustituyendo la lista vacía [] por la variable *interacciones* en los argumentos de la función *lm_stepwise* (Se ha optado usar solo el método stepwise para las interacciones por simplificar la extensión del documento). De nuevo, los valores obtenidos para los parámetros principales son similares a los modelos anteriores.

Una vez generados todos los modelos con cada método de selección de variables y tras analizar el número de grados de libertad y el valor de los coeficientes, sospechamos que algunos de estos modelos puedan ser idénticos entre sí. Para verificar esto, se ha añadido al conjunto de funciones propias *FuncionesMineria.py* la siguiente función:

```
def comparar_modelos(modelo1, modelo2):
    # Extraer los parámetros dependiendo del tipo de modelo
    def obtener_params(modelo):
        if hasattr(modelo['Modelo'], 'params'):  # statsmodels
            return modelo['Modelo'].params.sort_values().copy()
        elif hasattr(modelo['Modelo'], 'coef_'):  # sklearn
            coef = modelo['Modelo'].coef_.flatten()
            intercept = modelo['Modelo'].intercept_
            return np.append(intercept, coef) if intercept.size > 0 else coef
        else:
            raise ValueError("El modelo no es compatible o no tiene parámetros disponibles.")

    try:
        params1 = obtener_params(modelo1)
        params2 = obtener_params(modelo2)

        if len(params1) != len(params2):
            print('Los modelos son diferentes. Tienen diferentes grados de libertad.')
        else:
            if np.allclose(params1, params2):
                print('Los modelos son iguales')
            else:
                print('Los modelos son diferentes. Los valores de sus parámetros no coinciden.')
    except Exception as e:
        print(f'Error al comparar modelos: {e}')
```

Esta comprueba la igualdad entre los grados de libertad y los coeficientes de cada par de modelos suministrado como input. Tras la aplicación de dicha función podemos comprobar que se obtiene exactamente el mismo modelo, independientemente del método de selección de variables siempre que se use la misma métrica (AIC o BIC). Por ende, tenemos solo 4 modelos diferentes: modeloStepAIC, modeloStepBIC, modeloStepAIC_int y modeloStepBIC_int.

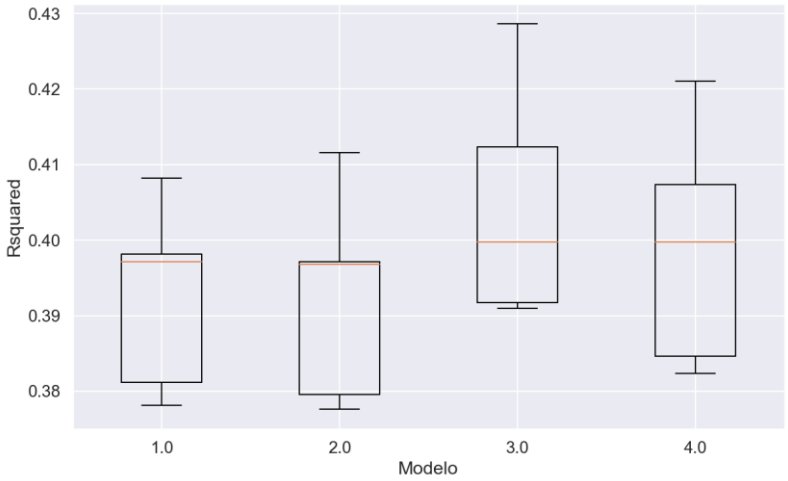
Centrándonos ahora en el coeficiente de correlación, aplicaremos una validación cruzada para diferentes particiones con el objetivo de elegir el mejor de los modelos entre los generados. Esto se hará a partir de las siguientes líneas de código:

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

for rep in range(20):

    modelo_stepAIC = validacion_cruzada_lm(
        5
        , x_train
        , y_train
        , modeloStepAIC['Variables']['cont']
        , modeloStepAIC['Variables']['categ']
    )
    modelo_stepBIC = validacion_cruzada_lm(
        5
        , x_train
        , y_train
        , modeloStepBIC['Variables']['cont']
        , modeloStepBIC['Variables']['categ']
    )
    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']
    )

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

Con los valores obtenidos del código anterior se hace una representación de cajas aplicando el siguiente código:

```
plt.figure(figsize=(10, 6))
plt.grid(True)
grupo_metrica = results.groupby('Modelo')['Rsquared']
boxplot_data = [grupo_metrica.get_group(grupo).tolist() for grupo in grupo_metrica.groups]
plt.boxplot(boxplot_data, labels=grupo_metrica.groups.keys())
plt.xlabel('Modelo')
plt.ylabel('Rsquared')
plt.show()
```

El resultado obtenido es el siguiente:



Numéricamente:

	MODELO 1 StepAIC	MODELO2 StepBIC	MODELO 3 StepAIC_int	MODELO 4 StepBIC_int
$MEAN(R^2)$	0.3925	0.3925	0.4047	0.3990
$STD(R^2)$	0.0113	0.0127	0.0143	0.0145
Freedom Degrees	57	55	64	73

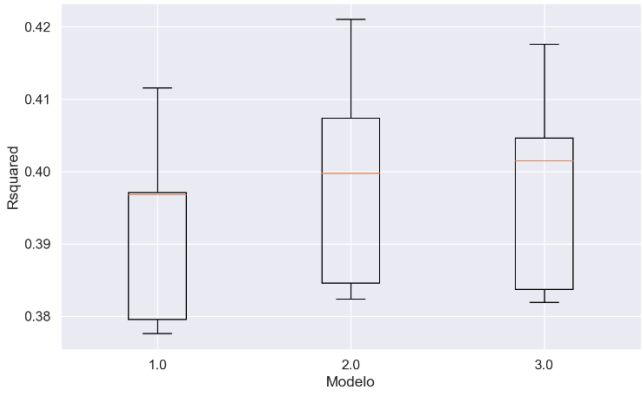
Nótese que tanto el valor medio como la desviación estándar para el coeficiente de correlación es bastante próximo en todos los casos. Por ende, atendiendo al principio de parsimonia, tomaremos como mejor modelo por el momento al modelo 2, este es, el *modeloStepBIC.*

4.2 Selección de variables aleatoria

Pasamos a continuación al método de selección de variables aleatoria. Para ello, aplicamos las siguientes líneas de código:

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

# Realizar 30 iteraciones de selección aleatoria.
for x in range(30):
    print('--------------------------- iter: ' + str(x))

    # Dividir los datos de entrenamiento en conjuntos de entrenamiento y prueba.
    x_train2, x_test2, y_train2, y_test2 = train_test_split(x_train, y_train,
                                                            test_size = 0.3,
                                                            random_state = 1234567 + x)

    # Realizar la selección stepwise utilizando el criterio BIC en la submuestra.
    modelo = lm_stepwise(y_train2.astype(int), x_train2, var_cont, var_categ,
                         interacciones, 'BIC')

    # Almacenar las variables seleccionadas y la fórmula correspondiente.
    variables_seleccionadas['Variables'].append(modelo['Variables'])
    variables_seleccionadas['Formula'].append(sorted(modelo['Modelo'].model.exog_names))

# Unir las variables en las fórmulas seleccionadas en una sola cadena.
variables_seleccionadas['Formula'] = list(map(lambda x: '+'.join(x),
                                              variables_seleccionadas['Formula']))

# Calcular la frecuencia de cada fórmula y ordenarlas por frecuencia.
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()

# Identificar las dos modelos más frecuentes y las variables correspondientes.
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])]
```

Una vez aplicado este proceso de selección aleatoria de dos conjuntos de variables, vamos a realizar una validación cruzada entre estos dos modelos obtenidos y el seleccionado como ganador en el apartado anterior (modeloStepBIC). No se mostrará el código por ser análogo al caso anterior. La representación de cajas y bigotes resulta:



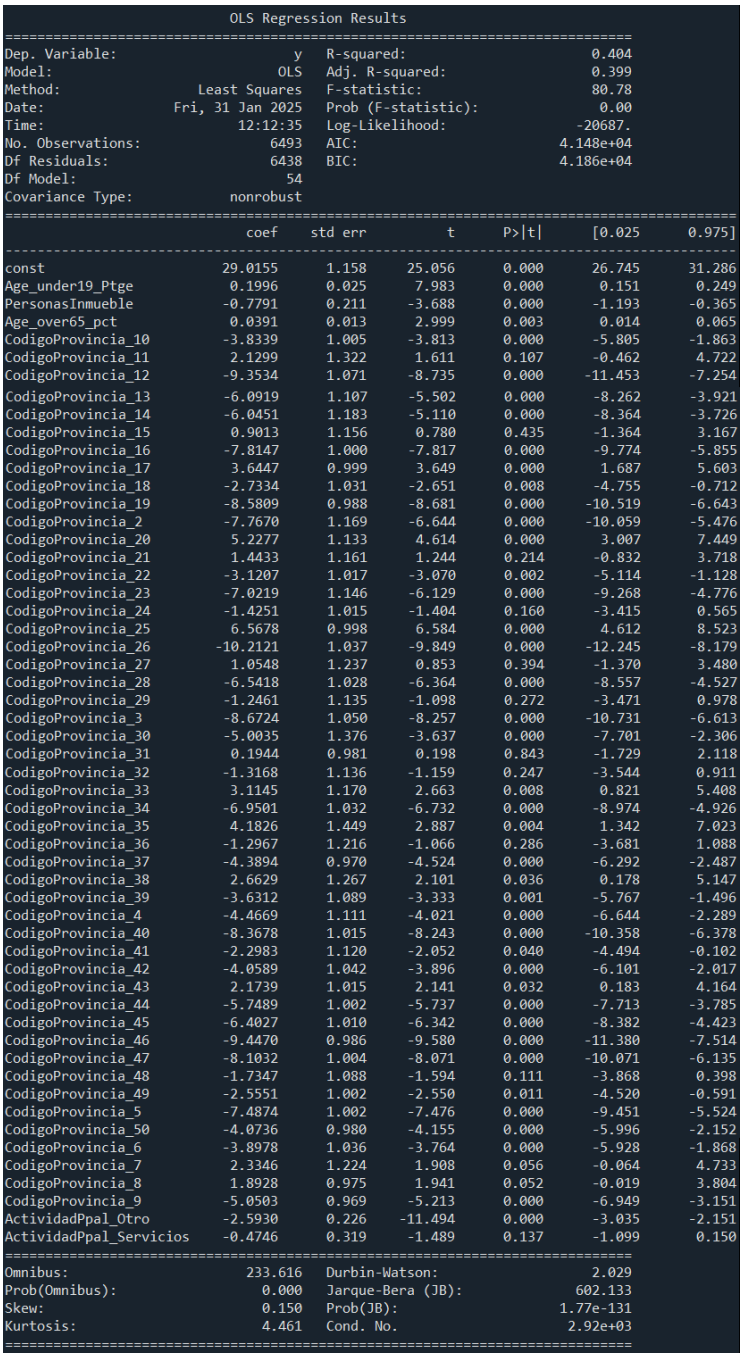
Numéricamente:

	MODELO 1 StepBIC	**MODELO2 Var_1**	**MODELO 3 Var_2**
$\boldsymbol{MEAN(R^2)}$	0.3925	0.3990	0.3979
$\boldsymbol{STD(R^2)}$	0.0127	0.0145	0.0135
Freedom Degrees	55	57	56

De nuevo, vemos que los valores medios y la desviación estándar de los coeficientes de correlación para los tres modelos son muy similares. Por ende, atendiendo de nuevo al principio de parsimonia, seleccionaremos como modelo ganador al modelo "*modeloStepBIC*".

4.3 Selección del modelo ganador

Tal y como se comentaba al final del apartado anterior, se ha seleccionado como modelo ganador al modelo "modeloStepBIC". El resumen de todos sus parámetros y coeficientes resulta ser:



OLS Regression Results

Dep. Variable:	y	R-squared:	0.404
Model:	OLS	Adj. R-squared:	0.399
Method:	Least Squares	F-statistic:	80.78
Date:	Fri, 31 Jan 2025	Prob (F-statistic):	0.00
Time:	12:12:35	Log-Likelihood:	-20687.
No. Observations:	6493	AIC:	4.148e+04
Df Residuals:	6438	BIC:	4.186e+04
Df Model:	54		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	29.0155	1.158	25.056	0.000	26.745	31.286
Age_under19_Ptge	0.1996	0.025	7.983	0.000	0.151	0.249
PersonasInmueble	-0.7791	0.211	-3.688	0.000	-1.193	-0.365
Age_over65_pct	0.0391	0.013	2.999	0.003	0.014	0.065
CodigoProvincia_10	-3.8339	1.005	-3.813	0.000	-5.805	-1.863
CodigoProvincia_11	2.1299	1.322	1.611	0.107	-0.462	4.722
CodigoProvincia_12	-9.3534	1.071	-8.735	0.000	-11.453	-7.254
CodigoProvincia_13	-6.0919	1.107	-5.502	0.000	-8.262	-3.921
CodigoProvincia_14	-6.0451	1.183	-5.110	0.000	-8.364	-3.726
CodigoProvincia_15	0.9013	1.156	0.780	0.435	-1.364	3.167
CodigoProvincia_16	-7.8147	1.000	-7.817	0.000	-9.774	-5.855
CodigoProvincia_17	3.6447	0.999	3.649	0.000	1.687	5.603
CodigoProvincia_18	-2.7334	1.031	-2.651	0.008	-4.755	-0.712
CodigoProvincia_19	-8.5809	0.988	-8.681	0.000	-10.519	-6.643
CodigoProvincia_2	-7.7670	1.169	-6.644	0.000	-10.059	-5.476
CodigoProvincia_20	5.2277	1.133	4.614	0.000	3.007	7.449
CodigoProvincia_21	1.4433	1.161	1.244	0.214	-0.832	3.718
CodigoProvincia_22	-3.1207	1.017	-3.070	0.002	-5.114	-1.128
CodigoProvincia_23	-7.0219	1.146	-6.129	0.000	-9.268	-4.776
CodigoProvincia_24	-1.4251	1.015	-1.404	0.160	-3.415	0.565
CodigoProvincia_25	6.5678	0.998	6.584	0.000	4.612	8.523
CodigoProvincia_26	-10.2121	1.037	-9.849	0.000	-12.245	-8.179
CodigoProvincia_27	1.0548	1.237	0.853	0.394	-1.370	3.480
CodigoProvincia_28	-6.5418	1.028	-6.364	0.000	-8.557	-4.527
CodigoProvincia_29	-1.2461	1.135	-1.098	0.272	-3.471	0.978
CodigoProvincia_3	-8.6724	1.050	-8.257	0.000	-10.731	-6.613
CodigoProvincia_30	-5.0035	1.376	-3.637	0.000	-7.701	-2.306
CodigoProvincia_31	0.1944	0.981	0.198	0.843	-1.729	2.118
CodigoProvincia_32	-1.3168	1.136	-1.159	0.247	-3.544	0.911
CodigoProvincia_33	3.1145	1.170	2.663	0.008	0.821	5.408
CodigoProvincia_34	-6.9501	1.032	-6.732	0.000	-8.974	-4.926
CodigoProvincia_35	4.1826	1.449	2.887	0.004	1.342	7.023
CodigoProvincia_36	-1.2967	1.216	-1.066	0.286	-3.681	1.088
CodigoProvincia_37	-4.3894	0.970	-4.524	0.000	-6.292	-2.487
CodigoProvincia_38	2.6629	1.267	2.101	0.036	0.178	5.147
CodigoProvincia_39	-3.6312	1.089	-3.333	0.001	-5.767	-1.496
CodigoProvincia_4	-4.4669	1.111	-4.021	0.000	-6.644	-2.289
CodigoProvincia_40	-8.3678	1.015	-8.243	0.000	-10.358	-6.378
CodigoProvincia_41	-2.2983	1.120	-2.052	0.040	-4.494	-0.102
CodigoProvincia_42	-4.0589	1.042	-3.896	0.000	-6.101	-2.017
CodigoProvincia_43	2.1739	1.015	2.141	0.032	0.183	4.164
CodigoProvincia_44	-5.7489	1.002	-5.737	0.000	-7.713	-3.785
CodigoProvincia_45	-6.4027	1.010	-6.342	0.000	-8.382	-4.423
CodigoProvincia_46	-9.4470	0.986	-9.580	0.000	-11.380	-7.514
CodigoProvincia_47	-8.1032	1.004	-8.071	0.000	-10.071	-6.135
CodigoProvincia_48	-1.7347	1.088	-1.594	0.111	-3.868	0.398
CodigoProvincia_49	-2.5551	1.002	-2.550	0.011	-4.520	-0.591
CodigoProvincia_5	-7.4874	1.002	-7.476	0.000	-9.451	-5.524
CodigoProvincia_50	-4.0736	0.980	-4.155	0.000	-5.996	-2.152
CodigoProvincia_6	-3.8978	1.036	-3.764	0.000	-5.928	-1.868
CodigoProvincia_7	2.3346	1.224	1.908	0.056	-0.064	4.733
CodigoProvincia_8	1.8928	0.975	1.941	0.052	-0.019	3.804
CodigoProvincia_9	-5.0503	0.969	-5.213	0.000	-6.949	-3.151
ActividadPpal_Otro	-2.5930	0.226	-11.494	0.000	-3.035	-2.151
ActividadPpal_Servicios	-0.4746	0.319	-1.489	0.137	-1.099	0.150

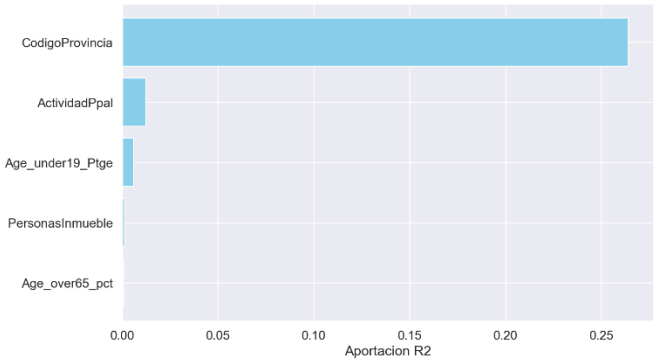
Omnibus:	233.616	Durbin-Watson:	2.029
Prob(Omnibus):	0.000	Jarque-Bera (JB):	602.133
Skew:	0.150	Prob(JB):	1.77e-131
Kurtosis:	4.461	Cond. No.	2.92e+03

Nótese que, por ejemplo, la variable continua "*Age_under19_Ptge*" tiene un coeficiente asociado con valor 0.1996. Esto implica que, si se mantienen constantes el resto de las variables del modelo y se aumenta en una unidad la variable "*Age_under19_Ptge*", la tasa de variación de la variable objetivo "AbstentionPtge" será del valor 0.1996. (Si "*Age_under19_Ptge*" aumenta un 1%, la variable "AbstentionPtge" aumenta $0.1996 \times 0.01 = 0.001996$ unidades).

La situación para las variables categóricas es ligeramente diferente. Recordamos que para variables de este tipo siempre existe una categoría que se toma como categoría de referencia (la primera de ellas por defecto). En el caso de la variable "CodigoProvincia" se toma como categoría de referencia "CodigoProvincia_1". Así, nótese que, por ejemplo, la categoría "CodigoProvincia_10" tiene un coeficiente asociado con valor -3.8339. Esto implica que, si se pretende analizar el valor de la variable objetivo "AbstentionPtge", este será -3.8339 veces

menor si el registro pertenece a la provincia 10 que si pertenece a la provincia 1 (categoría de referencia).

Por otra parte, si nos fijamos en los p-valores asociados a los diferentes coeficientes, vemos que prácticamente todos ellos son menores que 0.05, lo cual es indicativo de la significancia de esas variables para el modelo. No obstante, es importante destacar que algunas de ellas sí que presentan un valor elevado. Concretamente, algunas categorías de las variables "*CodigoProvincia*" y "*ActividadPpal*". Probablemente se deba a que estas categorías están poco representadas en el conjunto de datos y quizás deberían haberse agrupado para aumentar el número de registros. Sin embargo, al tratarse de provincias, no parece lógico agruparlas sin criterio alguno más que el de tener pocas observaciones. Aplicando la función propia *modelEffectSizes* a este modelo podemos apreciar que, en efecto, son estas dos variables las que mayor relevancia tienen sobre la correlación del modelo, en consonancia con lo mencionado:



Sin embargo, en líneas generales el modelo parece bastante robusto. Esto lo podemos observar por la poca diferencia entre los valores del coeficiente de correlación R^2 para los datos de entrenamiento y test de la partición inicial, que resultan 0.4039 y 0.3970 respectivamente.

5 Modelo de regresión logística

Para este apartado, comenzamos también realizando una partición de los datos de manera análoga a como se hizo en el modelo de regresión lineal. La única diferencia, exceptuando el cambio en la variable objetivo, será indicar que las variables dependientes serán valores enteros. Esto se hará con la siguiente línea de código:

```
y_train, y_test = y_train.astype(int), y_test.astype(int)
```

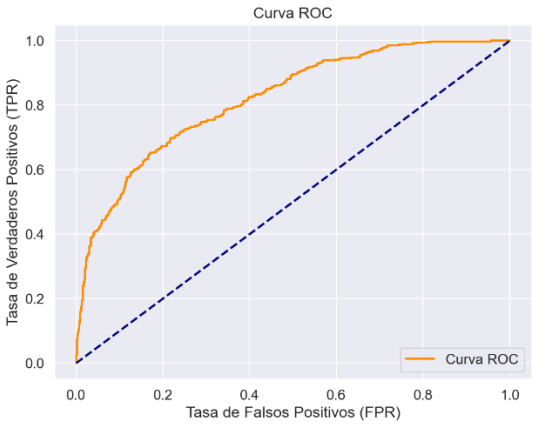
A continuación, se procederá con la aplicación de los diferentes métodos de selección de variables. En este caso no nos centraremos en los mismos parámetros que en el modelo de regresión lineal. Aquí los protagonistas para determinar la calidad de un modelo serán el parámetro *pseudo* R^2 y el *AUC-ROC.*

5.1 Selección de variables clásica

Tal y como se procedió en el desarrollo del modelo de regresión lineal, se mostrarán las líneas de código asociadas solo a uno de los métodos de selección clásica de variables (el resto son idénticos). Concretamente, para la aplicación del método *stepwise* con métrica AIC se tiene:

```
modeloStepAIC_glm = glm_stepwise(y_train, x_train, var_cont, var_categ, [], 'AIC')
summary_glm(modeloStepAIC_glm['Modelo'], y_train, modeloStepAIC_glm['X'])
pseudoR2(modeloStepAIC_glm['Modelo'], modeloStepAIC_glm['X'], y_train)
x_test_modeloStepAIC_glm = crear_data_modelo(x_test, modeloStepAIC_glm['Variables']['cont'],
                                             modeloStepAIC_glm['Variables']['categ'],
                                             modeloStepAIC_glm['Variables']['inter'])
pseudoR2(modeloStepAIC_glm['Modelo'], x_test_modeloStepAIC_glm, y_test)
```

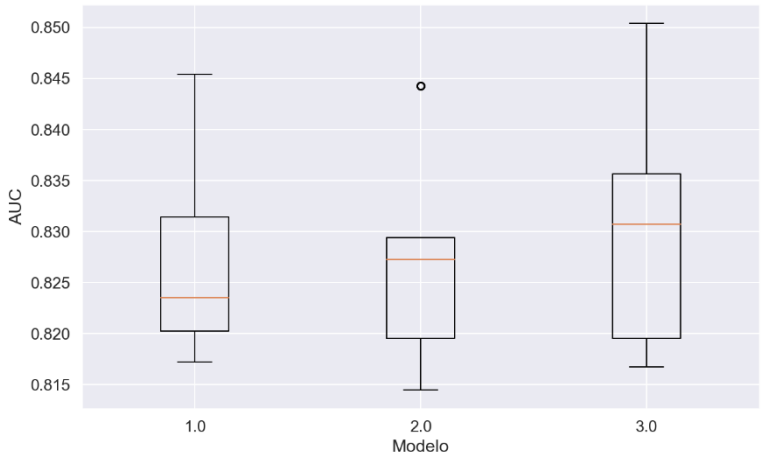
Tras la aplicación de este código se han obtenido valores del parámetro pseudo R^2 de 0.2813 para el conjunto de datos de entrenamiento y 0.2513 para el conjunto de datos de prueba. El valor en ambos casos refleja que el modelo tiene un desempeño bastante razonable. Además, la poca diferencia en el valor de este parámetro para los dos conjuntos de datos es un indicativo de la robustez del modelo. Por otra parte, también se ha representado la curva ROC y el área bajo la misma (usando la función propia curva_roc()). El resultado obtenido ha sido:



El valor obtenido para el área bajo dicha curva es de 0.8199, lo cual también es un buen indicativo de la efectividad de este modelo.

Tras la generación de los 6 modelos (stepwise, backward y forward con cada una de las métricas AIC o BIC), comprobamos que todos los modelos son diferentes con la misma función propia definida en el apartado de regresión lineal. El resultado que se obtiene es que los modelos generados con el mismo método de selección de variables son idénticos a pesar de la métrica empleada (AIC o BIC). Además, es importante destacar que en la implementación del método forward ha surgido un problema con la matriz de covarianzas. Esto puede deberse a multicolinealidad entre los datos o a un número de parámetros muy elevado en relación con la cantidad de datos, por lo que vamos a descartar los dos modelos generados con este método.

De nuevo en analogía con el apartado de regresión lineal, se generan dos modelos (stepwiseAIC y stepwiseBIC) pero con interacciones entre las variables continuas. Estos dos métodos han resultado idénticos entre sí. Por ende, resulta un total de 3 modelos diferentes que, al aplicarles la validación cruzada para el parámetro AUC, nos ofrecen el siguiente resultado (no se muestran las líneas de códigos por ser muy similares a las del capítulo anterior):



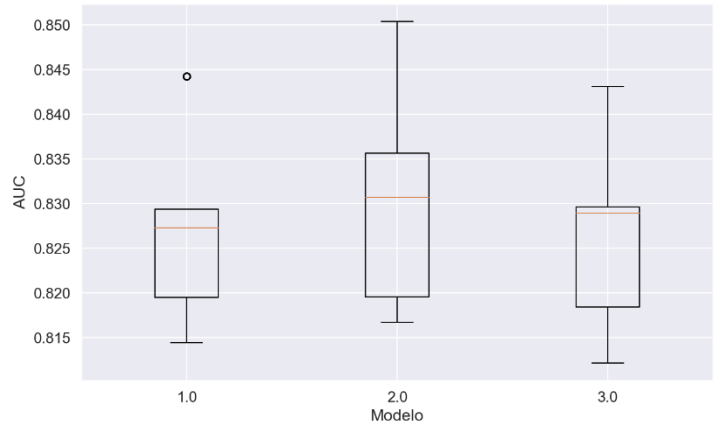
Numéricamente tendríamos:

		Modelo 1 Stepwise		Modelo 2 Backward		Modelo 3 Stepwise_int	
Mean(AUC)		0.8276		0.8270		0.8306	
Std(AUC)		0.0101		0.0102		0.0122	
Freedom Degrees		70		68		71	
Pseudo R^2		0.2813	0.2513	0.2795	0.2483	0.2861	0.2577
Train	**Test**						

Nótese que todos los modelos generados presentan un valor de AUC bastante similar. El valor del parámetro *pseudo* R^2 no se ha obtenido haciendo validación cruzada, sino únicamente para la partición inicial de los datos. Sin embargo, vemos que todos los valores son bastante próximos entre los datos *train* y los datos *test*, además de ser próximos entre todos los modelos. Por ello, atendiendo al menor valor de grados de libertad, consideraremos como ganador el modelo 2. Esto es, el modelo generado por el método de selección de variables Backward con métrica AIC.

5.2 Selección de variables aleatorias

En este apartado también procedemos de forma análoga a como se hizo en el capítulo referente al modelo de regresión lineal, así que tampoco se añadirán las líneas de código. Directamente, se procede a mostrar el resultado de la validación cruzada (centrada de nuevo en el parámetro AUC) para dos modelos generados con elección aleatoria de variables y el ganador del apartado inmediatamente anterior.



Numéricamente sería:

	Modelo 1 Backward	Modelo 2 Aleatorio 1	Modelo 3 Aleatorio 2
Mean(AUC)	0.8270	0.8306	0.8265
Std(AUC)	0.0102	0.0122	0.0107
Freedom Degrees	68	71	68

Se observa de nuevo una proximidad muy acentuada entre todos los valores (tanto medios como desviaciones asociadas). Por tanto, atendiendo de nuevo al menor número de grados de libertad y una mínima diferencia en el valor medio del parámetro AUC y su desviación estándar, consideramos de nuevo al modelo generado con el método Backward como el mejor modelo entre todos los generados.

5.3 Selección del modelo ganador

Ya en los apartados anteriores se ha concluido que el mejor modelo entre todos los generados parecer ser el que se ha construido con el método de selección clásica de variables Backward. En este capítulo se buscará el mejor punto de corte para este modelo, se presentarán algunos valores que justifiquen su fiabilidad y se discutirá la interpretación de algunos coeficientes de este.

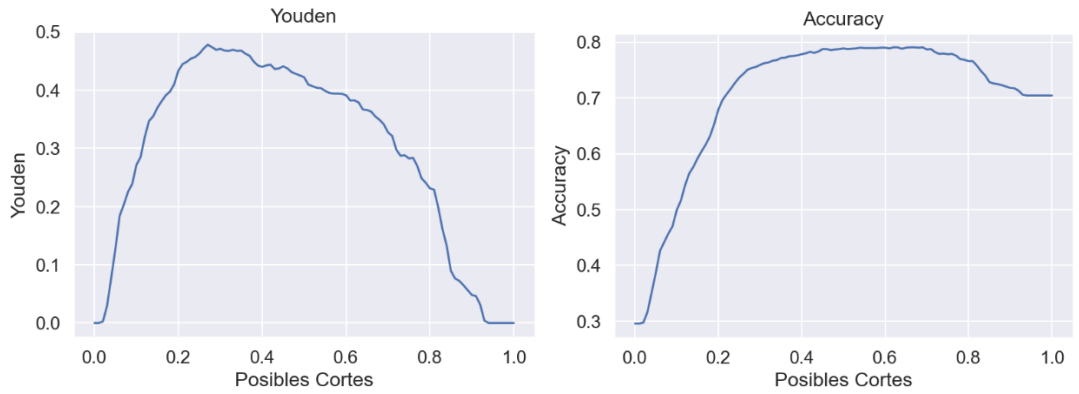
En primer lugar, en lo que respecta a la búsqueda del mejor punto de corte, se aplican las siguientes líneas de código:

```
posiblesCortes = np.arange(0, 1.01, 0.01).tolist()  # Generamos puntos de corte de 0 a 1 con intervalo de 0.01
rejilla = pd.DataFrame({
    'PtoCorte': [],
    'Accuracy': [],
    'Sensitivity': [],
    'Specificity': [],
    'PosPredValue': [],
    'NegPredValue': []
})  # Creamos un DataFrame para almacenar las métricas para cada punto de corte

for pto_corte in posiblesCortes:  # Iteramos sobre los puntos de corte
    rejilla = pd.concat(
        [rejilla, sensEspCorte(modeloBackAIC_glm['Modelo'], x_test, y_test, pto_corte,
                               modeloBackAIC_glm['Variables']['cont'],
                               modeloBackAIC_glm['Variables']['categ'],
                               modeloBackAIC_glm['Variables']['inter'])],
        axis=0
    )
```

Una vez realizado este mapeado, calculamos el índice de Youden con la siguiente línea de código:

```
rejilla['Youden'] = rejilla['Sensitivity'] + rejilla['Specificity'] - 1
```

Con todo lo anterior, vamos a analizar los valores de dicho índice y del parámetro "*Accuracy*" para los diferentes puntos de corte de la rejilla. Se obtuvieron las siguientes representaciones gráficas:



Concretamente, el valor máximo del índice de Youden se obtiene para el punto de corte 0.27, mientras que el valor máximo del parámetro "*Accuracy*" se obtiene para el punto de corte 0.62. A continuación, vamos a estudiar el resto de los parámetros para esos dos puntos de cortes. Esto se hará a partir de las siguientes líneas de código:

```
sensEspCorte(modeloBackAIC_glm['Modelo'], x_test, y_test, 0.27,
             modeloBackAIC_glm['Variables']['cont'],
             modeloBackAIC_glm['Variables']['categ'],
             modeloBackAIC_glm['Variables']['inter'])
sensEspCorte(modeloBackAIC_glm['Modelo'], x_test, y_test, 0.62,
             modeloBackAIC_glm['Variables']['cont'],
             modeloBackAIC_glm['Variables']['categ'],
             modeloBackAIC_glm['Variables']['inter'])
```

Los resultados obtenidos se engloban en la siguiente tabla:

Pto. Corte	Accuracy	Sensitivity	Specificity	PosPredValue	NegPredValue
0.27	0.7506	0.7104	0.7675	0.5618	0.8633
0.62	0.7906	0.4479	0.9344	0.7414	0.8013

Si nuestro objetivo fuese minimizar los falsos negativos, es clara que la elección correcta sería el punto 0.27 por tener una mayor sensibilidad. Sin embargo, como nuestro objetivo es maximizar el balance general del modelo, nos centraremos en aquel que tenga un mayor nivel de "*Accuracy*", lo cual hace que la elección del mejor punto de corte sea el 0.62.

Recordamos además que, para la partición inicial de los datos, los valores del parámetro pseudo R^2 para los datos de entrenamiento y los datos test fueron, respectivamente, 0.2795 y 0.2483. La poca diferencia entre ambos valores es un buen indicativo de la robustez del modelo.

Por otra parte, los valores de AUC para la misma partición de los datos mencionada resulta de 0.8336 y 0.8182 (train y test respectivamente). De nuevo, esta poca diferencia entre los dos conjuntos de datos, así como un valor cercano a 1, es un buen indicativo de la eficiencia del modelo.

Adicionalmente, se han comparado los valores de "*Accuracy*", "*Sensitivity*", "*Specificity*", etc, para la misma partición mencionada y con el punto de corte 0.62. El resultado se engloba en la siguiente tabla:

Datos	Accuracy	Sensitivity	Specificity	PosPredValue	NegPredValue
Train	0.7970	0.4834	0.9413	0.7912	0.7984
Test	0.7906	0.4479	0.9344	0.7414	0.8013

Una vez más, la proximidad de los valores para la distinta selección de los parámetros es una garantía de la robustez del modelo.

Para finalizar, se añadirá el valor de los coeficientes asociados a cada variable de este modelo, así como la posterior interpretación de algunos de ellos.

```
Variable: Age_over65_pct, Coeficiente: 0.01268375593853792
Variable: CCAA_Aragón, Coeficiente: -0.25569092310401326
Variable: CCAA_Asturias, Coeficiente: 0.900341778658643
Variable: CCAA_Baleares, Coeficiente: 0.668166854249053
Variable: CCAA_Canarias, Coeficiente: 0.8402004942467252
Variable: CCAA_Cantabria, Coeficiente: -0.4584356290258717
Variable: CCAA_CastillaLeón, Coeficiente: -0.9506533892152612
Variable: CCAA_CastillaMancha, Coeficiente: -1.1741413371191547
Variable: CCAA_Cataluña, Coeficiente: 1.7733104830719484
Variable: CCAA_ComValenciana, Coeficiente: -1.5936298587627977
Variable: CCAA_Extremadura, Coeficiente: -0.2873596393191303
Variable: CCAA_Galicia, Coeficiente: 0.3439190420794332
Variable: CCAA_Madrid, Coeficiente: -0.6404109011231248
Variable: CCAA_Murcia, Coeficiente: -0.2842912261827776
Variable: CCAA_Navarra, Coeficiente: 0.32306615295765595
Variable: CCAA_PaísVasco, Coeficiente: 1.0580858538625144
Variable: CCAA_Rioja, Coeficiente: -1.2152855840347654
Variable: CodigoProvincia_10, Coeficiente: -0.22322064511163706
Variable: CodigoProvincia_11, Coeficiente: 1.6534417408077091
Variable: CodigoProvincia_12, Coeficiente: -0.19597087256803136
Variable: CodigoProvincia_13, Coeficiente: -0.050339185519203605
Variable: CodigoProvincia_14, Coeficiente: -1.2556847256268748
Variable: CodigoProvincia_15, Coeficiente: 0.5882452830548708
Variable: CodigoProvincia_16, Coeficiente: -0.6398189688853415
Variable: CodigoProvincia_17, Coeficiente: 0.6200410822260491
Variable: CodigoProvincia_18, Coeficiente: 0.07950250138359632
Variable: CodigoProvincia_19, Coeficiente: -0.05410733840308918
Variable: CodigoProvincia_2, Coeficiente: -0.6099994384843573
Variable: CodigoProvincia_20, Coeficiente: 1.623979333277805
Variable: CodigoProvincia_21, Coeficiente: 1.109758098467411
Variable: CodigoProvincia_22, Coeficiente: 0.12834775254180428
Variable: CodigoProvincia_23, Coeficiente: -1.2141296905619534
Variable: CodigoProvincia_24, Coeficiente: 0.6214923449551613
Variable: CodigoProvincia_25, Coeficiente: 0.9726418336414202
Variable: CodigoProvincia_26, Coeficiente: -1.2152855840347654
Variable: CodigoProvincia_27, Coeficiente: 0.23873497298937343
Variable: CodigoProvincia_28, Coeficiente: -0.6404109011231248
Variable: CodigoProvincia_29, Coeficiente: 0.6544701597603241
Variable: CodigoProvincia_3, Coeficiente: -0.2598128835346489
Variable: CodigoProvincia_30, Coeficiente: -0.2842912261827776
Variable: CodigoProvincia_31, Coeficiente: 0.32306615295765595
Variable: CodigoProvincia_32, Coeficiente: -0.3553999427841027
Variable: CodigoProvincia_33, Coeficiente: 0.900341778658643
Variable: CodigoProvincia_34, Coeficiente: -0.30860643892200423
Variable: CodigoProvincia_35, Coeficiente: 0.16627567860208442
Variable: CodigoProvincia_36, Coeficiente: -0.12766127118070733
Variable: CodigoProvincia_37, Coeficiente: 0.23244478824360207
Variable: CodigoProvincia_38, Coeficiente: 0.6739248156446414
Variable: CodigoProvincia_39, Coeficiente: -0.4584356290258717
Variable: CodigoProvincia_4, Coeficiente: -0.36899145714911813
Variable: CodigoProvincia_40, Coeficiente: -1.056628250674859
Variable: CodigoProvincia_41, Coeficiente: 0.2939957168557735
Variable: CodigoProvincia_42, Coeficiente: 0.4667587309090272
Variable: CodigoProvincia_43, Coeficiente: -0.014263204155060672
Variable: CodigoProvincia_44, Coeficiente: -0.1825852767679278
Variable: CodigoProvincia_45, Coeficiente: 0.1801235941728357
Variable: CodigoProvincia_46, Coeficiente: -1.1378461026601168
Variable: CodigoProvincia_47, Coeficiente: -1.3934163097104424
Variable: CodigoProvincia_48, Coeficiente: -0.7158696417248591
Variable: CodigoProvincia_49, Coeficiente: 0.8102057703838675
Variable: CodigoProvincia_5, Coeficiente: -0.35487503097207107
Variable: CodigoProvincia_50, Coeficiente: -0.2014533988778888
Variable: CodigoProvincia_6, Coeficiente: -0.06413899420749353
Variable: CodigoProvincia_7, Coeficiente: 0.668166854249053
Variable: CodigoProvincia_8, Coeficiente: 0.19489077135954144
Variable: CodigoProvincia_9, Coeficiente: 0.03197100657245828
Variable: ActividadPpal_Otro, Coeficiente: -0.9541289958975081
Variable: ActividadPpal_Servicios, Coeficiente: -0.17460729481304804
```

Nótese que, por ejemplo, la variable numérica “Age_over65_pct” tiene un coeficiente asociado con valor 0.0127. Este coeficiente refleja el cambio en el log-odds de obtener un positivo en la variable objetivo. Dicho de otro modo, un aumento del 1% en la variable “Age_over65_pct” aumenta la probabilidad un $e^{0.0127} \approx 1.0128\%$ de obtener el valor 1 en la variable dicotómica objetivo.

Por otra parte, si atendemos por ejemplo a la categoría “CCAA_Aragón” de la variable categórica “CCAA”, vemos que tiene un coeficiente asociado con valor -0.2557. De forma análoga al caso de la regresión lineal, aquí esta categoría actúa como variable dummy. Por ende, el significado de este parámetro habla de cuánto disminuye (en este caso disminuye por ser un coeficiente negativo) la probabilidad de obtener un 1 en la variable dicotómica objetivo si el registro es en Aragón en comparación con si el registro es en la CCAA de referencia.

6 Conclusiones

El presente trabajo ha permitido explorar y aplicar técnicas de minería de datos y modelización predictiva en el contexto de la abstención electoral en municipios de España. A través de un proceso estructurado que incluyó la depuración de datos, la selección de variables y la implementación de modelos de regresión lineal y logística, se lograron identificar factores clave que influyen en la participación ciudadana en las elecciones municipales.

En primer lugar, la fase de limpieza y depuración de datos fue fundamental para garantizar la calidad del conjunto de datos. Se corrigieron errores en la asignación de tipos de variables, se identificaron y trataron valores atípicos, y se implementaron estrategias para la imputación de valores faltantes (este último punto podría discutirse en mayor detalle). Este proceso permitió reducir la distorsión en los análisis posteriores y mejorar la fiabilidad de los modelos generados.

En cuanto a la modelización, el análisis de correlaciones permitió seleccionar un subconjunto de variables relevantes, destacando aquellas relacionadas con la composición demográfica y la actividad socioeconómica de los municipios. El modelo de regresión lineal mostró una relación significativa entre la abstención y variables como la proporción de población joven y el número de empresas en el municipio. No obstante, se evidenció la presencia de problemas de normalidad en los residuos, lo cual sugiere que hubiese sido interesante estudiar transformaciones adicionales sobre las variables explicativas (aunque esto no era objeto de estudio para este trabajo).

Por su parte, el modelo de regresión logística permitió predecir con mayor precisión si la abstención superará el 30%, obteniendo un área bajo la curva ROC de aproximadamente 0.83. Esto indica una capacidad discriminatoria alta del modelo. La selección del punto de corte óptimo, basada en el índice de Youden y la métrica de precisión, permitió maximizar el balance entre sensibilidad y especificidad. Se observó que factores como la edad de la población y la comunidad autónoma de pertenencia tienen una influencia significativa en la abstención electoral.

En términos generales, los modelos desarrollados presentan una buena capacidad predictiva y permiten extraer conclusiones valiosas sobre los determinantes de la abstención en las elecciones municipales en España.