IMF Smart Education

MÁSTER EN DATA SCIENCE & BUSINESS ANALYTICS ONLINE

LOCALIZACIÓN DE ZONAS GEOGRÁFICAS CON ALTO POTENCIAL DE MEJORA EN EFICIENCIA ENERGÉTICA

TFM elaborado por: Carlos González-Novo Martín
Tutor de TFM: Abel Soriano Vázquez

- Madrid, 19 de noviembre de 2023 -

Índice

Lista de figuras

Lista de tablas

Resumen

Este estudio se concentra en analizar minuciosamente los certificados energéticos en la región de Extremadura, empleando datos del catastro, de la junta de Extremadura, del Instituto Nacional de Estadística y del Centro Nacional en Innovación Geográfica, los cuales son procesados utilizando técnicas de ciencia de datos. El enfoque principal es agrupar los certificados en categorías mediante un modelo de superposición basado en las coordenadas geográficas de los edificios. Se busca desarrollar un proyecto integral con un panel de control que permita a los usuarios explorar un mapa digital y visualizar distintas agrupaciones de edificios diferenciadas por el certificado energético más predominante.

Adicionalmente, se plantea un segundo objetivo ambicioso: realizar un análisis predictivo para clasificar la eficiencia energética de los edificios utilizando datos catastrales, coordenadas geográficas, zona climática y renta neta promedio por distrito. No obstante, este objetivo no se logra debido a la escasa cantidad y variedad de registros de viviendas con certificados energéticos, así como a la falta de variables con correlación significativa con la variable objetivo, entre las que se han evaluado.

Las aplicaciones potenciales del proyecto incluyen agilizar la toma de decisiones en políticas energéticas en Extremadura, revitalizar áreas con baja calificación energética y promover el desarrollo sostenible, beneficiando tanto a los propietarios como a la comunidad en general.

1. Estado del arte

1.1 Catastro inmobiliario

El Catastro Inmobiliario es un registro administrativo, de carácter oficial, bajo la jurisdicción del gobierno de España y adscrito al Ministerio de Hacienda y Función Pública. Su principal objetivo radica en llevar a cabo un minucioso inventario y descripción de los bienes inmuebles, abarcando tanto aquellos de naturaleza rústica como urbana, e incluso aquellos que presenten características especiales.

Esta importante herramienta se encuentra regulada por el Texto Refundido de la Ley del Catastro Inmobiliario (Real decreto administrativo 1/2004), el cual establece de manera categórica la obligatoriedad de la inscripción en dicho catastro, sin que esto implique ningún tipo de carga económica, lo cual lo diferencia sustancialmente del Registro de la Propiedad.

La descripción catastral de los bienes inmuebles comprende una amplia gama de aspectos que abarcan tanto sus características físicas, como las económicas y jurídicas. Dentro de estas, se incluyen elementos tales como:

- Localización y referencia catastral precisa.
- Superficie total del inmueble.
- Uso o destino asignado.
- Clase de cultivo o aprovechamiento.
- Calidad de las construcciones.
- Representación gráfica.
- Valor de referencia de mercado.
- Valor catastral.
- Identificación precisa del titular catastral, incluyendo su número de identificación fiscal o número de identidad en caso de extranjeros.
- Coordinación con el Registro de la Propiedad, en caso de existir, junto con el código registral asignado.

1.2 Certificados energéticos de las edificaciones

El 1 de junio del 2021 se aprobó el procedimiento básico para la certificación de la eficiencia energética de los edificios de España (BOE-A-2021-9176), con el objetivo de fomentar la eficiencia energética y promover el ahorro de energía en el sector de la edificación.

El decreto establece que todos los edificios nuevos, así como aquellos que sean objeto de venta o alquiler, deben contar con un certificado de eficiencia energética. Este certificado proporciona información sobre el consumo energético del edificio y su impacto en el medio ambiente, asignándole una calificación energética que va desde la letra A (más eficiente) hasta la letra G (menos eficiente).

Figura 1. *Modelo de certificación de eficiencia energética del edificio existente* [3]

Esta calificación energética es la expresión de la eficiencia energética de un edificio, o parte de este, determinada según la metodología establecida en el documento reconocido correspondiente al Procedimiento básico, y se expresa mediante indicadores energéticos y la etiqueta de eficiencia energética, acorde a la definición descrita en el (BOE-A-2021-9176).

En el mismo documento se establecen distintas categorías de certificación de eficiencia energética, así como otras definiciones relevantes. A continuación, se presenta una tabla que resume estas definiciones de manera visual:

Concepto	Definición
Certificación de eficiencia energética de proyecto	Proceso de valoración de la calificación de la eficiencia energética de edificios nuevos, reformas o ampliaciones, basado en las características especificadas en el proyecto. Conduce a la expedición del certificado de eficiencia energética de proyecto.
Certificación de eficiencia energética de obra terminada	Proceso de valoración de la calificación de la eficiencia energética de edificios nuevos, reformas o ampliaciones, basado en las características reales del edificio u obra terminada. Permite comparar con la calificación obtenida en la certificación de eficiencia energética de proyecto. Conduce a la expedición del certificado de eficiencia energética de obra terminada.
Certificación de eficiencia energética de edificio existente o parte de este	Proceso de valoración de la calificación de eficiencia energética obtenida a partir de datos calculados o medidos del edificio existente o parte de él. Conduce a la expedición del certificado de eficiencia energética del edificio existente.
Certificado de eficiencia energética de proyecto	Documento suscrito por el técnico competente como resultado del proceso de certificación. Contiene información sobre las características energéticas, la calificación de eficiencia energética del proyecto y recomendaciones de posibles intervenciones para mejorar la eficiencia energética de cada edificio o parte de este.
Certificado de eficiencia energética de obra terminada	Documento suscrito por el técnico competente como resultado del proceso de certificación. Incluye información sobre las características energéticas, la calificación de eficiencia energética y recomendaciones de posibles intervenciones para mejorar la eficiencia energética de un edificio nuevo, reforma o ampliación. Permite comparar la calificación obtenida en la certificación de eficiencia energética de proyecto.
Certificado de eficiencia energética de edificio existente	Documento suscrito por el técnico competente que contiene información sobre las características energéticas, la calificación de eficiencia energética y recomendaciones de posibles intervenciones para mejorar la eficiencia energética de un edificio existente o parte del mismo.

Tabla 1. *Definiciones certificaciones energéticas* (BOE-A-2021-9176)

Se sugiere adquirir una mayor comprensión del tema mediante la consulta exhaustiva del Real Decreto 390/2021, así como considerar el documento de preguntas frecuentes disponible (FAQs RD 390/2021), el cual brinda aclaraciones sobre diversos conceptos presentes en este Real Decreto.

1.3 AsistenteCDT (Extremadura)

El asistente para la Confección de Documentación Técnica (CDT) o también llamado AGILE es un sistema informático creado para facilitar la confección de las Carpetas Técnicas (CT) según lo establecido en el Decreto 115/2018 [5], que regula la certificación de eficiencia energética de edificios en la Comunidad Autónoma de Extremadura y crea el Registro de Certificaciones de Eficiencia Energética de Edificios. Estas CT contienen la documentación técnica requerida para la inscripción de certificados de eficiencia energética.

El proceso de confección de las CT se realiza electrónicamente por los técnicos competentes, quienes cargan el informe de evaluación energética del edificio en formato electrónico (XML) y completan el formulario básico conforme al anexo I del Decreto 115/2018.

Dentro del Registro de Certificaciones de la Eficiencia Energética de Edificios de Extremadura, están publicados un total de 45.520 registros con la siguiente información:

- Dirección
- Código de referencia de la herramienta.
- Categoría del consumo energético.
- Categoría de Emisiones de CO2.
- Referencia catastral.

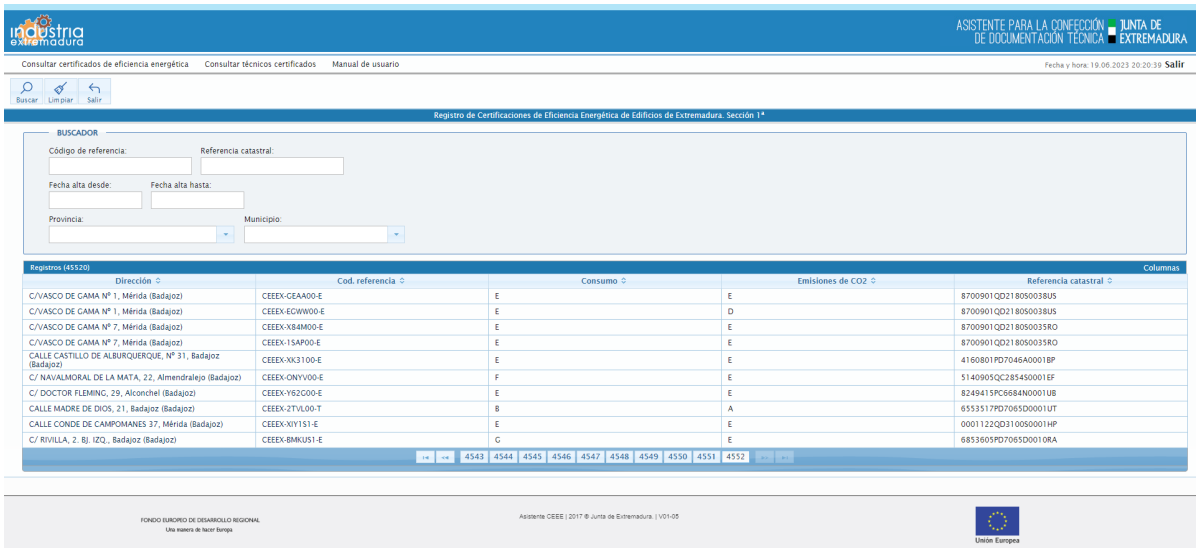


Figura 2. *Configuración de la herramienta AGILE*

Además, accediendo individualmente a cada certificado se puede llegar a extraer los datos de la evaluación energética del edificio en formato PDF.

Para llegar a comprender la codificación implementada en los datos, la Junta de Extremadura proporciona un manual de usuario del Asistente para la confección de documentación técnica (Consejería para la Transición Ecológica y Sostenibilidad).

1.4 ArcGIS

ArcGIS es un sistema de información geográfica (SIG) desarrollado por la empresa Esri. Es ampliamente utilizado en una variedad de campos, desde la planificación urbana y la gestión ambiental hasta la respuesta a desastres y la toma de decisiones empresariales basadas en la ubicación. ArcGIS permite crear, visualizar, analizar y compartir datos espaciales, lo que lo convierte en una herramienta esencial para comprender patrones geográficos y tomar decisiones informadas. [7]

Las características principales de ArcGIS incluyen:

- **Visualización de datos espaciales.** ArcGIS permite crear mapas y visualizar datos en formas gráficas y comprensibles. Esto es crucial para comprender patrones y relaciones geográficas.
- **Análisis espacial.** Ofrece una amplia gama de herramientas para realizar análisis complejos en datos espaciales. Esto incluye operaciones como geoprocesamiento, análisis de proximidad, interpolación, análisis de rutas, análisis de patrones, etc.
- **Gestión de datos.** Proporciona capacidades para crear, editar y gestionar datos geográficos. Puede integrar datos de múltiples fuentes, mantener su calidad y asegurarse de que sean precisos y actualizados.
- **Integración de datos.** Puede trabajar con datos de diversos formatos y fuentes, desde imágenes satelitales hasta bases de datos, permitiendo combinar información de diferentes fuentes para un análisis más completo.
- **Creación de mapas temáticos.** Permite la creación de mapas con temáticas específicas para representar datos complejos de manera visualmente comprensible. Esto es valioso para la presentación de información a diversos públicos.
- **Herramientas de edición.** Ofrece herramientas para editar y manipular elementos en los mapas, como agregar capas, cambiar estilos y etiquetado, y dibujar formas.
- **Compartir y colaborar.** Permite compartir mapas y datos con otros usuarios.
- **Aplicaciones personalizadas.** Se pueden crear aplicaciones específicas basadas en la plataforma ArcGIS para satisfacer necesidades específicas de análisis y visualización de datos geoespaciales.

- **Soporte para datos 3D:** ArcGIS admite la representación y análisis de datos en tres dimensiones, lo que es valioso para aplicaciones como modelado de terreno, planificación urbana y análisis de edificios.
- **Integración con otras Herramientas:** Puede integrarse con otras herramientas de análisis, como R y Python, para un procesamiento y análisis más avanzado.

En resumen, ArcGIS es una plataforma integral para la gestión y análisis de datos geográficos, que juega un papel crucial en muchas industrias y disciplinas para la toma de decisiones informadas basadas en la ubicación.

2. Objetivos del proyecto

El objetivo principal de este estudio es analizar detalladamente los certificados energéticos en la región de Extremadura. Para lograrlo, se emplearán datos del catastro y de la junta de Extremadura, que se analizarán mediante técnicas de ciencia de datos. Una vez obtenidos los resultados, se busca analizar y gestionar los datos geográficos mediante la plataforma ArcGIS. En este proceso, se agruparán los certificados en sus categorías mediante un modelo de superposición basado en las coordenadas geográficas de los edificios y las áreas de agrupación definidas. Por ello, se implementará un mapa digital introducido en un panel de control interactivo, que ofrecerá diferentes formas de agrupar la información. Esta herramienta se usará para presentar visualmente Indicadores Clave de Desempeño o KPI, por sus siglas en inglés, que se actualizarán a medida que el usuario actúe a través de un mapa digital.

Además, se plantea un segundo objetivo más amplio: realizar un análisis predictivo para clasificar la eficiencia energética de edificios basándose en datos catastrales, coordenadas geográficas, zona climática y renta neta promedio por distrito. Esto permitirá predecir la eficiencia energética de edificios donde esta información es desconocida, lo que aumentará el número de registros de certificación y apoyará el objetivo principal del proyecto.

Alcanzar los objetivos daría lugar a una herramienta integral y versátil, con la capacidad de agilizar la toma de decisiones en políticas energéticas en Extremadura, revitalizar áreas con baja calificación energética y promover el desarrollo sostenible, beneficiando tanto a los propietarios como a la comunidad en general.

Para ello, se propone seguir los siguientes pasos:

1. Identificación de los datos:
    a. Descarga de la información catastral de la provincia. [8]
    b. Recopilación automática de datos del registro oficial de Certificaciones de Eficiencia Energética de Edificios de Extremadura mediante la técnica de *web scrapping*. [9]
2. Limpieza y preprocesado de datos:
    a. Limpieza de las dos fuentes de datos, aplicando métodos de estandarización, transformación de formatos, imputación de valores faltantes, homogeneización de datos, correlación, mapeo y detección de errores, etc.
    b. Preparación de los datos para facilitar el modelado de estos.
3. Análisis y modelado:
    a. Planteamiento de un modelo predictivo adecuado.

a. Validación y pruebas del modelo.

    i. Robustez del modelo.

    ii. Capacidad de generalización del modelo.

    iii. Características de ejecución del modelo.

4. Aplicación del modelo.

5. Presentación de resultados ArcGIS.

    a. Aplicación de modelos de agrupación espacial en ArcGIS.

    b. Creación del dashboard.

3. Material y modelos

Esta sección ofrece una visión detallada de los materiales y modelos utilizados en el proyecto, desde la extracción y tratamiento de los datos hasta la creación la configuración de los modelos aplicados.

3.1 Extracción de los datos catastrales

Catastro inmobiliario gestiona una plataforma en línea conocida como la Sede Electrónica de Catastro (SEC). Esta plataforma digital ofrecida por Catastro brinda acceso a los datos catastrales de todas las áreas bajo su jurisdicción, que abarca la totalidad del territorio nacional, excluyendo País Vasco y Navarra.

En primer lugar, la SEC permite la descarga de información catastral de las propiedades (excepto detalles de propiedad y valor catastral). Esta información está organizada por municipios y los archivos de datos se encuentran disponibles en diferentes formatos, los cuales se describen en los enlaces de asistencia. Utilizar la plataforma requiere que los usuarios se autentiquen mediante certificado electrónico o Cl@ve, además de aceptar los términos de una licencia de uso para acceder a dicha información.

Indicar que la fecha de descarga de estos ficheros aparece en el nombre del propio fichero, y que la información de estos ficheros se actualiza dos veces al año, a principios de febrero y agosto.

3.1.1 Descarga y procesamiento de la información alfanumérica

Los datos con la información alfanumérica de las zonas urbanas se extraen en formato CAT, comprimidos previamente como archivos GZ para reducir el tamaño de la descarga, y agrupados por provincias en formato ZIP. Por lo tanto, para su tratamiento es imprescindible descomprimirlos e ingeniar un código que permita hacer una lectura de la información. El código utilizado para descomprimir se encuentra inscrito en el archivo "0_Descomprimir_archivos_gz.py".

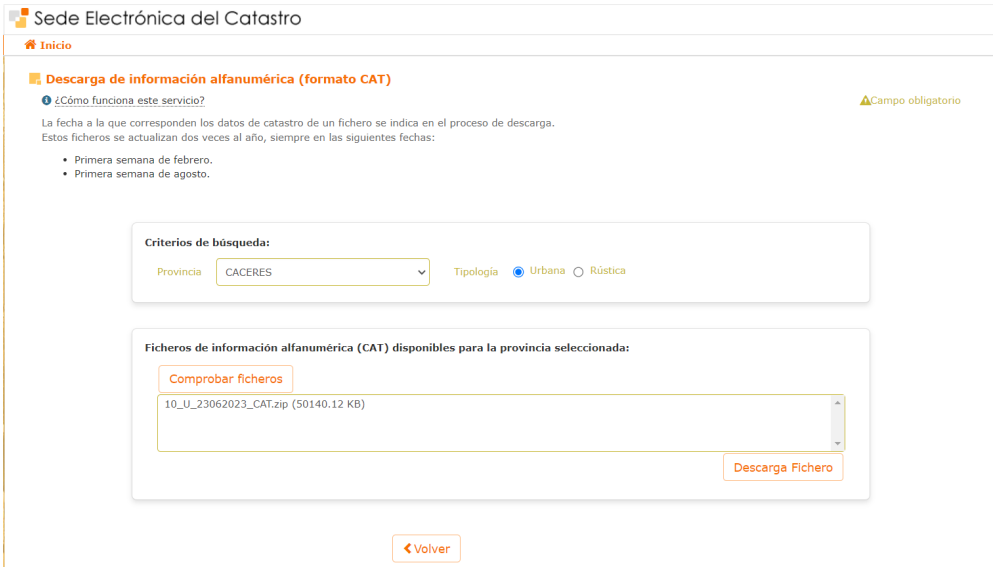


Figura 3. *Panel de descarga de los archivos CAT*

Los archivos descomprimidos siguen una estructura de nombre similar a la siguiente: “06136U_05012023.CAT”. Los primeros dos dígitos corresponden a la codificación de la provincia del INE, los tres siguientes al código del municipio al que está vinculado en la base de datos catastral. La letra indica el tipo de datos (U: Urbanos y R: Rústicos), y los últimos dígitos representan la fecha en que se descargó el archivo.

Los ficheros CAT almacenan la información de los registros catastrales dividida en tipologías. La descripción de cada tipología se trata a continuación:

- **Tipo 01: Registro de cabecera.** Existe uno para todo el fichero independientemente de que el fichero recoja la información correspondiente a un solo municipio o a varios.
- **Tipo 11: Registro de Finca.** Existe uno por cada parcela catastral implicada. Identifica y localiza a la parcela catastral.
- **Tipo 13: Registro de Unidad Constructiva.** Existe uno por cada unidad constructiva en cada parcela catastral. Representa un edificio o un conjunto de construcciones particularizadas dentro de un edificio.
- **Tipo 14: Registro de Construcción.** Existe uno por cada construcción de cada unidad constructiva en cada parcela catastral. Identifica cada uno de los locales existentes en un bien inmueble, con su descripción física: superficie, antigüedad, tipología.
- **Tipo 15: Registro de Inmueble.** Existirá uno por cada bien inmueble en cada parcela catastral. Identifica cada uno de los bienes inmuebles dentro de una parcela catastral. Reparto de elementos comunes.
- **Tipo 16: Registro de reparto de elementos comunes.** Existirá al menos uno por cada elemento común que se reparte, siempre que sea necesario especificar repartos especiales. Identifica el elemento constructivo cuyo valor se reparte entre los demás elementos de construcción.
- **Tipo 17: Registro de cultivos.** Existirá uno por cada subparcela de cultivo existente dentro de la parcela catastral. Identifica cada subparcela de cultivo existente dentro de la parcela catastral.
- **Tipo 90: Registro de cola.** Existirá uno para todo el fichero.

Tal como se ha mencionado previamente, un registro puede presentar diversas tipologías y, de cada una de ellas, es posible extraer información alfanumérica distinta.

Catastro proporciona una guía detallada que describe la estructura de la información de acuerdo con cada tipología [10]. Esta guía facilita la posibilidad de extraer información de manera precisa y eficiente en un proceso masivo utilizando el lenguaje de programación Python.

A pesar de que Catastro también ofrece plantillas para el tratamiento de archivos en formatos como Excel y OpenOffice, es importante tener en cuenta que la extracción de datos a través de estos métodos puede ser laboriosa y propensa a cometer errores. Por esta razón, se ha desarrollado el código en Python descrito en el archivo "1_Data_Catastro.py".

La información extraída para cada registro catastral es la siguiente: referencia catastral, provincia, municipio, distrito, código postal, dirección, escalera, planta, puerta, dimensiones de la construcción, tipo de inmueble, tipo de uso del inmueble, año de reforma (en caso de no haberse reformado el edificio corresponde al año de construcción), superficie de la finca y las coordenadas del centroide de la parcela.

A continuación, en orden cronológico, se describen las características de los archivos PY creados para la extracción de la información catastral de las viviendas:

1. **0_Descomprimir_archivos_gz.** Como se había comentado anteriormente es necesario descomprimir de manera masiva los archivos descargados de Catastro para extraer la información alfanumérica de las viviendas. Por ello, previamente se requiere realizar una descompresión de los archivos ZIP; y a continuación, una segunda de los archivos GZ. Para ello, es necesario consultar la literatura de las librerías *os*, *gzip* y *zipfile*.

   Se decide utilizar la librería *os* ya que, proporciona una interfaz para interactuar con el sistema operativo subyacente. De esta manera se pueden extraer las rutas de los archivos que se quieran descomprimir de forma masiva.

   Por otro lado, dado que únicamente se requieren descomprimir dos archivos ZIP, correspondientes a las provincias de Extremadura, no es necesario utilizar la librería *zipfile*, pues de forma manual se pueden extraer de manera sencilla.

   Ahora bien, la situación se complica con la extracción de los archivos GZ, ya que la propia librería *gzip* a pesar de contener funciones como `gzip.compress()` o `gzip.decompress()`, produce ciertos errores de compilación en el caso de querer descomprimir los archivos en memoria. Por ello, es necesario almacenarlos descomprimidos en formato CAT en una ubicación permanente. Para ello, se crean dos definiciones: `write_file(filename, data, out_path)` y `decompress(filename, in_path, out_path)`.

```python
# Definiciones

def write_file(filename, data, out_path):
    try:
        f = open(os.path.join(out_path, filename), "wb")
    except IOError as e:
        print(e.errno, e.message)
    else:
        f.write(data)
        f.close()


def decompress(filename, in_path, out_path):
    with gzip.open(os.path.join(in_path, filename), 'rb') as f_in:
        write_file(filename[:filename.rfind(".gz")], f_in.read(), out_path)
```

Figura 4. *Definiciones para descomprimir archivos GZ*

La función `write_file` está diseñada para escribir datos en un archivo binario en una ruta especificada. Primero, intenta abrir un archivo en modo binario para escritura en la ruta proporcionada. Si la operación tiene éxito, los datos se escriben en el archivo y luego se cierra. En caso de que ocurra un error de E/S, como se indica mediante la excepción `IOError`, la función imprime el número de error y el mensaje correspondiente.

La función `decompress` se encarga de descomprimir archivos GZ. Utiliza `gzip.open` para abrir el archivo comprimido en modo lectura binaria, lee los datos comprimidos y luego llama a la función `write_file` para escribir los datos descomprimidos en un nuevo archivo en la ruta de salida. El nombre del archivo descomprimido se genera eliminando la extensión GZ del nombre del archivo original. En conjunto, estas funciones permiten la descompresión y escritura de archivos comprimidos de manera controlada y gestionan posibles errores durante el proceso.

Con estas definiciones implementadas, el proceso de descompresión de archivos GZ se simplifica notablemente. Al emplear un par de bucles y aprovechar las funciones proporcionadas, como `os.listdir()` para obtener las direcciones de los archivos a descomprimir, y `decompress` para extraer los archivos CAT en la ubicación designada, es posible realizar la descompresión de los 388 elementos resultantes de manera eficiente y rápida. Se recomienda consultar el código en el caso de no estar familiarizado con las funciones aquí mencionadas.

2. **1_Data_Catastro.** Este código en Python aborda de manera integral el proceso de lectura y procesamiento de los archivos de Catastro, los cuales se presentan en formato CAT tras ser descomprimidos. La estructura del código refleja una meticulosa planificación para gestionar grandes volúmenes de datos de manera eficiente y estructurada.

   La primera sección del código se encarga de importar las librerías necesarias para el manejo de datos *(`pandas`),* la interacción con el sistema operativo (`os`), la medición del tiempo de ejecución (`time`), y

las transformaciones de coordenadas geográficas (*pyproj*). Además, se establecen diccionarios que actuarán como referencias para mapear códigos específicos a descripciones, facilitando así la interpretación de los datos.

A continuación, se definen las rutas de los directorios de entrada y salida. La lista "list_archives" se crea para contener las rutas completas de los archivos en el directorio de entrada. Paralelamente, se inicializan dataframes vacíos para albergar la información de inmuebles, geolocalización y construcción, creando una estructura organizada para el almacenamiento de datos procesados.

Es esencial destacar que el procesamiento de la información alfanumérica requiere un tiempo de ejecución que supera las 8 horas. Ante esta consideración, se introducen variables como *i*, "num_inm", "num_geoloc" y "num_constr" antes del inicio del bucle. Esta medida permite la inicialización del código desde un archivo CAT específico. En consecuencia, en caso de interrupciones, no es necesario reiniciar el pretratamiento de los datos desde el principio, optimizando así la eficiencia y manejabilidad del proceso.

Seguidamente, se ingresa a un bucle principal que itera sobre los archivos en "list_archives". En cada ciclo, realiza la lectura del contenido del archivo, generando dataframes que contienen información específica sobre inmuebles, geolocalización y construcción (información correspondiente a las tipologías 11, 14 y 15). Se aplican transformaciones detalladas a las coordenadas geográficas, y se implementa una lógica de almacenamiento en archivos CSV cada vez que la longitud de los dataframes alcanza 25.000 filas.

Uno de los motivos fundamentales para seleccionar un límite de almacenamiento en memoria de 25.000 filas está relacionado con el uso de la librería *pandas*. A medida que el conjunto de dataframes aumenta, los procesos de cálculo asociados a estos dataframes experimentan una ralentización. Por consiguiente, al verificar que con una cantidad de 25.000 registros apenas se observa una reducción significativa en el tiempo de ejecución, se opta por este valor como un compromiso eficiente en términos de rendimiento y gestión de recursos.

Finalmente, con el fin de asegurar que todos los datos son guardados, los últimos dataframes se guardan en archivos CSV independientemente del número de filas que contengan, brindando un cierre ordenado al proceso de procesamiento.

En la siguiente figura se presenta el procedimiento de extracción de información de coordenadas geográficas almacenada en formato CAT. Este proceso se destaca por su enfoque minucioso en la manipulación y transformación de datos geográficos. Además, se lleva a cabo la extracción de información alfanumérica pertinente contenida en archivos CAT del tipo 11. Este análisis abarca desde la definición de segmentos clave en el conjunto de datos original hasta la creación de un dataframe

especializado, denominado *"*df_geoloc*"*. Este dataframe resultante incorpora tanto la información geoespacial transformada como otros datos alfanuméricos relevantes, proporcionando así una representación más completa y utilizable de la información contenida en los archivos CAT tipo 11.

```
#############
# DF_GEOLOC
#############

# Definir las rebanadas de cadenas para cada columna del df_geoloc
slices = {
    'tipo': (0, 2),
    'cod_parc_cat': (30, 44),
    'nom_vp_sig': (158, 163),
    'nom_vp': (163, 188),
    'num_1_pol': (188, 192),
    'nom_1_letra': (192, 193),
    'num_2_pol': (193, 197),
    'nom_2_letra': (197, 198),
    'x': (333, 342),
    'y': (342, 352),
    'nom_EPSG': (666, 676)
}

df_geoloc_i = pd.DataFrame(data)

# Extraer los valores de cada columna utilizando rebanadas de cadenas
for col_name, (start, end) in slices.items():
    df_geoloc_i[col_name] = df_geoloc_i[0].str[start:end]

df_geoloc_i = df_geoloc_i[df_geoloc_i['tipo'] == '11']

# Transformación de coordenadas
df_geoloc_i = df_geoloc_i[~(df_geoloc_i['nom_EPSG'] == '0000000000')]
df_geoloc_i.reset_index(drop=True, inplace=True)

df_geoloc_i['x'] = df_geoloc_i['x'].astype(float) / 100
df_geoloc_i['y'] = df_geoloc_i['y'].astype(float) / 100

# Definir el sistema de coordenadas de destino (WGS84 latitud-longitud)
crs_destino = pyproj.CRS.from_string('EPSG:4326')

# Aplicar la transformación a las coordenadas 'x' y 'y' utilizando apply y lambda
df_geoloc_i['longitud'], df_geoloc_i['latitud'] = zip(*df_geoloc_i.apply(
    lambda row: pyproj.Transformer.from_crs(
        pyproj.CRS.from_string(row['nom_EPSG']),
        crs_destino,
        always_xy=True
    ).transform(row['x'], row['y']),
    axis=1
))

# Eliminar columnas innecesarias
df_geoloc_i.drop(columns=[0, 'tipo', 'x', 'y', 'nom_EPSG'], inplace=True)
df_geoloc = pd.concat([df_geoloc, df_geoloc_i], ignore_index=True)
```

Figura 5. *Extracción de la información del dataframe df_geoloc*

En este fragmento de código, se definen segmentos, denominados "slices", que actúan como indicadores de las posiciones en el conjunto de datos original. Estos segmentos delinean los límites de campos clave, como el tipo, el código de parcela catastral, la dirección de la parcela, los números de

polígono, las coordenadas “x” e “y”, y el sistema de referencia EPSG de estas coordenadas. Esta estrategia de segmentación establece una estructura esencial para organizar la información en columnas específicas dentro del dataframe resultante, denominado “df_geoloc”.

A continuación, se crea un dataframe temporal, “df_geoloc_i”, utilizando el conjunto de datos original. La extracción de valores para cada columna se realiza mediante el uso de las rebanadas de cadenas previamente definidas, contribuyendo a la organización y estructuración adecuada de la información.

En el siguiente paso, se filtran las filas del dataframe temporal para incluir únicamente aquellas que poseen el tipo '11', indicando así la presencia de información geográfica relevante. Luego, se inicia un proceso de transformación de coordenadas, eliminando las filas con un código EPSG de '0000000000' (registros sin coordenadas). Las coordenadas “x” e “y” se convierten a valores flotantes y se dividen por 100 para ajustarlas al formato adecuado. Esto es debido a que la información de las coordenadas se extrae con dos decimales, pero sin el separador decimal; y que la longitud de sus caracteres es de 9 y 10, respectivamente.

Cada registro contiene información sobre el huso geográfico SRS de sus coordenadas. Por tanto, se declara la definición del sistema de coordenadas de destino mediante la especificación del código EPSG 4326, correspondiente a WGS84 (latitud-longitud), y el sistema de coordenadas de origen. La aplicación de esta transformación a las coordenadas “x” e “y” se realiza de manera eficiente mediante el uso de la función `apply` y `lambda`, asegurando una conversión precisa para cada fila del dataframe. Esta conversión se almacena en las dos nuevas columnas llamadas longitud y latitud.

Finalmente, se eliminan columnas innecesarias, como el tipo, las coordenadas originales “x” e “y”, y el código EPSG. El dataframe temporal (“df_geoloc_i”,) se concatena con el dataframe principal (“df_geoloc”), garantizando la integridad del índice mediante la opción `ignore_index=True`. En conjunto, este bloque de código demuestra un enfoque sistemático y meticuloso para la manipulación de datos geográficos. Además, su estructura es muy similar a la utilizada para extraer la información de los registros tipo 14 y 15. Se recomienda consultar el código para una mayor comprensión de este.

3. **2_Unificar_csv_y_limpieza_df.** Este código en Python aborda de manera integral el proceso de unificación de archivos CSV, se busca obtener una única estructura de datos que contenga todos los registros con código de destino de la construcción de vivienda (V). Este filtrado es esencial para minimizar la presencia de valores vacíos en el archivo resultante del código. Sin embargo, surge un inconveniente: al cruzar estos datos con los certificados de eficiencia energética de Extremadura, se puede encontrar que los certificados de eficiencia energética de proyecto no estén presentes en este conjunto de datos. Esto se debe a que Catastro probablemente considere a la parcela con un código de destino distinto al de vivienda. Además, en el caso de viviendas construidas recientemente, los

parámetros de registro podrían no estar actualizados, lo que podría resultar en la pérdida de algunos registros. Estas consideraciones motivan la incorporación del segundo objetivo del proyecto.

Respecto a la estructura del código, se comienza con la importación las librerías necesarias (*pandas* para la manipulación y análisis de datos, *os* para interactuar con el sistema operativo, *numpy* para realizar operaciones numéricas eficientes y *warnings* para manejar advertencias durante la ejecución del código), configurando algunas opciones de visualización (`pd.options.display.max_columns = None` para mostrar en la consola todas las columnas de los dataframes, y `warnings.filterwarnings('ignore')` para ignorar los mensajes de advertencia que podrían ser generados por el intérprete de Python durante la ejecución del programa). Luego, se definen funciones, como `cont_palabras` y `describe_columnas`, que están diseñadas para analizar la frecuencia de palabras y describir columnas específicas de un dataframe, respectivamente. Funciones muy útiles a la hora de normalizar y limpiar los datos tras su unificación.

A continuación, se establecen los directorios para los archivos CSV que se utilizarán y se procede a cargar y unir estos archivos. Se distinguen tres tipos de archivos: construcción, geolocalización e inmuebles. Estos se unen en tres dataframes: *df_catastro_con*, *df_catastro_loc*, y *df_catastro_inm*. Seguidamente, se limpian los valores no numéricos en columnas específicas de cada dataframe, eliminando espacios en blanco y reemplazando valores vacíos con `None`.

Es importante considerar que estos tres dataframes no contienen ninguna columna común con la que se pueda fusionar la información en un único archivo de datos. Por ello, es necesario generar esta columna en cada uno de los dataframes. En concreto, se utiliza la referencia catastral de 20 caracteres para la unión de la información de los archivos construcción e inmueble. *Y* el código de parcela catastral (14 caracteres) para la fusión de los datos de los archivos de geolocalización con el resultante.

Notesé, que realizar la unión por medio de una referencia catastral carente del código de control (18 caracteres) lleva a realizar un cruce de información errónea entre los archivos de construcción e inmuebles. Sin embargo, la fusión de datos de las coordenadas se puede realizar conociendo únicamente el código de la parcela catastral ya que, este valor es único para cada parcela catastral independientemente de las construcciones e inmuebles en su interior. Se recuerda que las coordenadas indicadas en Catastro corresponden con el centroide de la parcela.

que realizar una unión a partir de los que existen duplicados en el *df_catastro_inm* y, por lo tanto, no se puede utilizar las funciones `merge` o `join` de manera que hay que mapear el dataframe para cruzar la información.

El código continúa, realizando una limpieza del dataframe *df_catastro_con* eliminando registros que contienen valores nulos en la columna “num_cargo” ya que, estos registros no aportan información

relevante. Luego, en el dataframe *df_catastro_inm*, se crea un código de referencia catastral combinando las columnas “cod_parc_cat”, “num_cargo” y “cod_control”. Esta nueva columna no es aún válida para realizar la unión de información del dataframe *df_catastro_inm* con el *df_catastro_con*, pues se desconoce el código de control (en esta situación la referencia catastral se compone únicamente de 18 caracteres).

A continuación, se preparan las variables del dataframe *df_catastro_inm* para localizar la referencia catastral de 20 caracteres de cada registro del dataframe *df_catastro_con*. Para ello, se alinean los valores en *df_catastro_inm* y *df_catastro_con*. Se identifican valores en las columnas “bloque”, “escalera”, “planta”, y “puerta” en *df_catastro_inm* que no están presentes en *df_catastro_con*, y se reemplazan con `None` en *df_catastro_inm*.

Adicionalmente, se realizan ajustes específicos en *df_catastro_inm*, como la sustitución de ciertos valores (“T” en “escalera”, “OD” en “planta”, “OS” en “puerta”) con `None`. Se establece un diccionario de orden de nexo y se crea una nueva columna “nexo” en *df_catastro_inm* que contiene listas de valores [1, 1, 1, 1, 1, 1]. Estas listas se modifican en función de ciertos criterios, filtrando y ajustando los datos según la presencia de valores nulos en “bloque”, “escalera”, “planta”, o “puerta”. De esta manera se genera una lista binaria donde el 0 indica que se trata de un valor nulo, y la posición de cada carácter hace referencia a una columna concreta (“dic_orden_nexo”). Seguidamente, a las columnas con valores no nulos se les asigna su nombre correspondiente, dando lugar a la columna “nexo_columnas”.

```
dic_orden_nexo = {
    0: 'cod_parc_cat',
    1: 'num_cargo',
    2: 'bloque',
    3: 'escalera',
    4: 'planta',
    5: 'puerta'}

df_catastro_inm['nexo'] =  [[1, 1, 1, 1, 1, 1]] * df_catastro_inm.shape[0]

# Filtrar
# filtro_num_cargo = df_catastro_inm[df_catastro_inm['cod_parc_cat'].isin(df_catastro_con[df_catastro_con['num_cargo'].isnull()]['cod_parc_cat'].tolist())].index
filtro_bloque = df_catastro_inm[df_catastro_inm['bloque'].isnull()]['bloque'].index
filtro_escalera = df_catastro_inm[df_catastro_inm['escalera'].isnull()]['escalera'].index
filtro_planta = df_catastro_inm[df_catastro_inm['planta'].isnull()]['planta'].index
filtro_puerta = df_catastro_inm[df_catastro_inm['puerta'].isnull()]['puerta'].index

# Aplicar una función lambda para modificar la lista 'nexo' en las filas filtradas
# df_catastro_inm.loc[filtro_num_cargo, 'nexo'] = df_catastro_inm.loc[filtro_num_cargo, 'nexo'].apply(lambda x: [x[0], 0, x[2], x[3], x[4], x[5]])
df_catastro_inm.loc[filtro_bloque, 'nexo'] = df_catastro_inm.loc[filtro_bloque, 'nexo'].apply(lambda x: [x[0], x[1], 0, x[3], x[4], x[5]])
df_catastro_inm.loc[filtro_escalera, 'nexo'] = df_catastro_inm.loc[filtro_escalera, 'nexo'].apply(lambda x: [x[0], x[1], x[2], 0, x[4], x[5]])
df_catastro_inm.loc[filtro_planta, 'nexo'] = df_catastro_inm.loc[filtro_planta, 'nexo'].apply(lambda x: [x[0], x[1], x[2], x[3], 0, x[5]])
df_catastro_inm.loc[filtro_puerta, 'nexo'] = df_catastro_inm.loc[filtro_puerta, 'nexo'].apply(lambda x: [x[0], x[1], x[2], x[3], x[4], 0])


# Obtener columnas a tener en cuenta según el filtro 'nexo'
df_catastro_inm['nexo_columnas'] = df_catastro_inm['nexo'].apply(lambda x: [dic_orden_nexo[idx] for idx, val in enumerate(x) if val == 1])

df_catastro_con.reset_index(drop=True, inplace=True)
df_catastro_inm.reset_index(drop=True, inplace=True)

# Crear una copia de df_temp
df_temp = df_catastro_con.copy()
```

***Figura* 6.** Creación de la columna “nexo_columnas” de *df_catastro_inm*

Esta nueva columna generada es la clave para localizar con precisión y exactitud la referencia catastral ("ref_cat") de *df_catastro_con* en *df_catastro_inm*. Se utilizarán las columnas indicadas en la columna "nexo_columnas" para determinar la referencia catastral de los registros de *df_catastro_con*. Por ende, la solución propuesta implica crea un diccionario que mapea los valores de la columna "ref_cat" en *df_catastro_inm* a través de combinaciones de columnas clave específicas de la variable "nexo_columnas". Luego, estos mapeos se aplican al dataframe con la información de las construcciones (*df_temp*) para crear nuevas columnas con un formato específico, que se utilizarán más adelante en el proceso de fusión.

```
################################################
# JOIN --> df_temp + df_catastro_inm = df_catastro

# Dado que tengo duplicados en el df_catastro_inm en función de las columnas claves, no puedo utilizar merge ni join por eso hay que mapear el df
# Crear un diccionario que mapee los valores de 'ref_cat' en df_catastro_inm
cuenta = 1
for columns in sorted(np.unique(df_catastro_inm['nexo_columnas'].values), key=lambda x: len(x)):
    print(cuenta,'-', columns, len(columns))

    mapping_dict_col = dict(zip(df_catastro_inm.set_index(columns).index, df_catastro_inm.set_index(columns)['ref_cat']))
    df_temp['ref_cat'] = df_temp.set_index(columns).index.map(mapping_dict_col)
    df_temp = df_temp.rename(columns={'ref_cat': 'ref_cat' + '_'+str(cuenta)})
    cuenta += 1

df_temp_2 = df_temp.copy()
# Se comprueba con la web de catastro que los registros con valores 'ref_cat_1' nulos son erróneos, ya que al buscar por su cod_parc_cat, la descripción no concuerda.
del_index = df_temp_2[df_temp_2['ref_cat_1'].isnull()].index
df_temp_2 = df_temp_2.drop(del_index)
# Obtener las columnas con el prefijo 'ref_cat'
list_col = [col for col in df_temp.columns if col.startswith('ref_cat')]
# Seleccionar el último valor no nulo de las columnas
df_temp_2['ref_cat'] = df_temp_2.apply(lambda row: row[list_col].dropna().iloc[-1], axis=1)
df_catastro = pd.concat([df_catastro_con.drop(del_index), df_temp_2['ref_cat']], axis = 1)
```

Figura 7. *Mapeo de los valores de "ref_cat" en df_catastro_inm y reasignación de "ref_cat"*

En el código, se utiliza un bucle `for` para iterar sobre combinaciones únicas de columnas clave en *df_catastro_inm*. La variable cuenta se introduce para numerar las nuevas columnas que se generarán en *df_temp*. Durante cada iteración, se crea un diccionario (*mapping_dict_col*) que relaciona los índices generados a partir de las columnas clave con los valores correspondientes de "ref_cat" en *df_catastro_inm*.

La aplicación de este mapeo se lleva a cabo en el dataframe *df_temp*, generando nuevas columnas denominadas "ref_cat_" seguidas del número de cuenta asignado.

Posteriormente, se procede a la eliminación de registros erróneos en "df_temp_2" identificados por valores nulos en la columna "ref_cat_1". Estos registros no concuerdan con la información extraída de la web del catastro y, por lo tanto, son eliminados.

Para la creación de la columna "ref_cat" final se selecciona el último valor no nulo de las columnas con el prefijo "ref_cat" en "df_temp_2" ya que, coincide con la referencia catastral asignada durante el mapeo de las columnas indicadas en "nexo_columnas" de cada registro. A continuación, se realiza la concatenación de *df_catastro_con* con `df_temp_2['ref_cat']` para construir el dataframe

definitivo, *df_catastro*. Durante este proceso, se eliminan los registros previamente identificados como erróneos.

Una vez conocidas las referencias catastrales de la columna "ref_cat" se identifican registros especiales en *df_catastro_inm* mediante la búsqueda de duplicados en la columna "ref_cat". Estos registros son críticos ya que, podrían causar problemas durante la fusión con los datos de localización de las parcelas. En concreto estos registros, cuentan con la característica de tener varias coordenadas y parámetros para una misma referencia catastral. Por lo tanto, en *df_catastro*, se seleccionan registros específicos asociados a los valores de "ref_cat": "000800100QD00G0001RQ", "000900100TK54C0001OO" y "002300800QD00H0001BP".

La fusión entre *df_catastro* y *df_catastro_inm* se realiza utilizando la columna "ref_cat" como clave y empleando una fusión de tipo izquierda (`how='left'`). Esto permite enriquecer *df_catastro* con información adicional proveniente de *df_catastro_inm*.

A continuación, se identifican los índices de los registros especiales en *df_catastro* y se procede a eliminar los duplicados tras la unión, conservando aquellos con información válida (índices 0, 3, 5 y 6 de *duplicated_values*). Se lleva a cabo la eliminación de columnas innecesarias generadas durante la fusión, específicamente aquellas que terminan con el sufijo "_y". Simultáneamente, se ajustan las columnas que contienen el sufijo "_x" eliminando dicho sufijo para una presentación más clara y concisa.

Para finalizar con esta sección del código, se realiza un restablecimiento del índice de *df_catastro* para garantizar coherencia en la estructura del dataframe resultante después de todas las operaciones de tratamiento y fusión de datos.

Llegados a este punto, se lleva a cabo el proceso de unión entre el dataframe *df_catastro* y *df_catastro_loc*. Primero, se crea una copia de *df_catastro* llamado *df_temp*. Luego, se identifican las columnas que existen en *df_catastro_loc* pero no en *df_catastro*. Para cada conjunto de columnas especificado en la variable *columns_to_join*, se crea un diccionario que mapea las claves correspondientes a esas columnas en *df_catastro_loc* a los valores de las columnas a unir en *df_temp*. Cada columna se renombra agregando un sufijo numérico a su nombre original (proceso similar al visto en la figura 7).

Posteriormente, se selecciona el último valor no nulo de las columnas con el prefijo "longitud" en *df_temp* y se crea una nueva columna llamada "nexo_columns_to_join". Esta columna almacena información sobre cuáles columnas se unieron en cada fila.

Se crea un nuevo dataframe, *df_temp_4*, que se genera a partir de aplicar una función lambda que extrae los valores de las columnas especificadas en "nexo_columns_to_join" para cada fila de *df_temp*. Este nuevo dataframe se concatena con *df_catastro*, ampliando así la información del Catastro con datos de localización.

A continuación, inicia la fase de limpieza de datos. Se procede a leer el archivo PARQUET, previamente almacenado, que contiene toda la información alfanumérica de Catastro en un único archivo ("Data_Catastro_brutos.parquet") y se ejecutan diversas operaciones de depuración. Aunque, en caso de ejecutar el código desde el principio, este paso podría considerarse prescindible, se opta por llevarlo a cabo. La razón radica en que se han efectuado múltiples análisis y se han aplicado diversas metodologías para unir y depurar los datos antes de alcanzar la configuración óptima. Por lo tanto, se prefiere mantener la coherencia en el flujo del proceso.

En esta etapa, se eliminan columnas que dejan de ser necesarias, tales como "cod_parc_cat", "num_cargo", "cod_control", "cod_dest", "nexo", y "nexo_columnas". Asimismo, se realizan ajustes en la representación de algunas variables, siendo notables las modificaciones que completan con ceros a la izquierda en las columnas "cod_prov_INE", "cod_mun_INE", y "num_dist".

Un cambio crucial se lleva a cabo en la variable "num_año", la cual se recalcula basándose en "num_año_reforma" solo si esta última no es igual a cero. Este proceso contribuye a garantizar una consistencia y coherencia en la representación de la antigüedad del inmueble, incorporando la información más precisa disponible en el dataset.

La limpieza de los registros de "escalera", "planta" y "puerta" se realiza mediante la sustitución de valores específicos y la normalización de ciertos caracteres y códigos.

Finalmente, se lleva a cabo un análisis descriptivo de las variables numéricas y se muestra información sobre tipos de datos y valores nulos en el dataframe resultante. El dataframe limpio se guarda en un archivo PARQUET denominado "Data_Catastro.parquet".

Por último, he de destacar que existe una fuente de información que aporta la resolución de las preguntas más frecuentes de estos archivos CAT [11]. Esta documentación a facilitado el entendimiento de algunas de las variables utilizadas en la construcción de "Data_Catastro.parquet".

3.1.2 Descarga de cartografía vectorial

La información geográfica en forma de cartografía vectorial se puede obtener en el formato Shapefile (SHP), lo cual facilita la integración de las capas catastrales de cada vivienda en ArcGIS.

La entidad de Catastro facilita un manual de usuario para descargar los archivos [12] y otro que describe la información de estos [13].

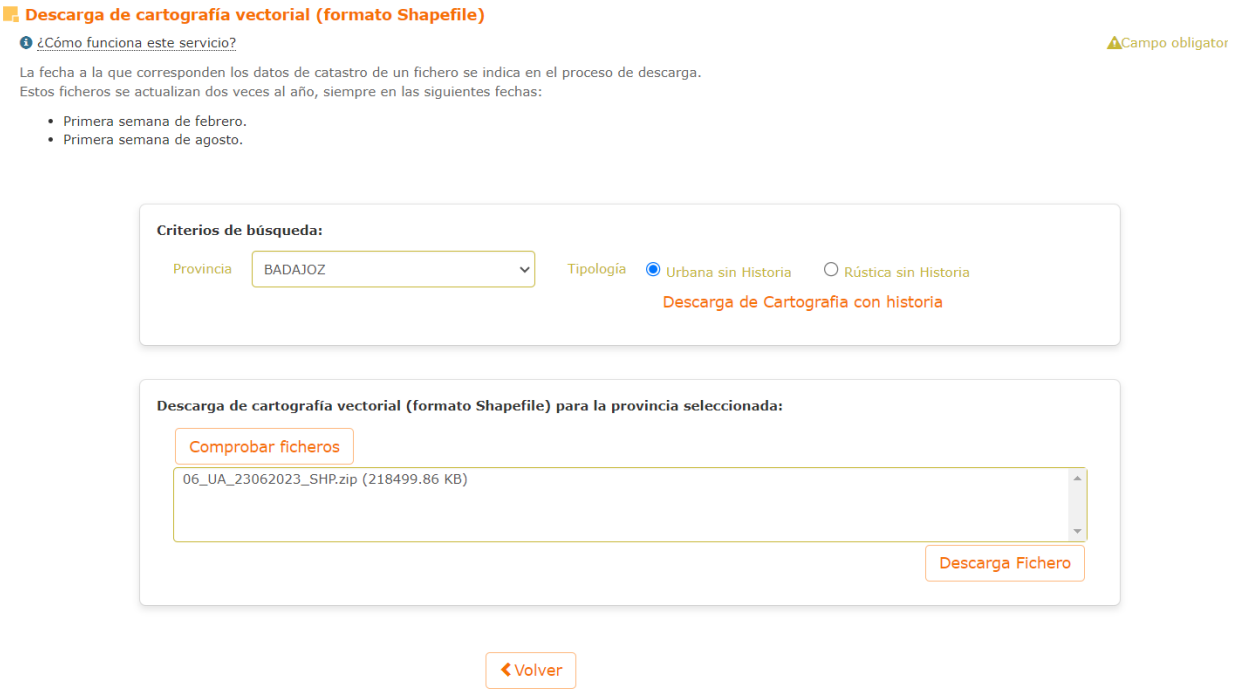


Figura 8. *Panel de descarga de los archivos SHP*

La información geométrica facilitada por la Sede Electrónica del Catastro contiene las siguientes tablas:

1. **ALTIPUN (Puntos de altimetría y redes geodésicas/topográficas).** Se refiere a puntos en un mapa que indican elevaciones o alturas del terreno, y también puntos de referencia de redes geodésicas y topográficas.
2. **CARVÍA (Descripción de códigos de vías e hidrografía).** Proporciona información sobre las diferentes carreteras, caminos u otras vías de comunicación, así como características relacionadas con cursos de agua.
3. **CONSTRU (Subparcelas urbanas que representan edificaciones).** Representa las áreas urbanas divididas en subparcelas que muestran las edificaciones y estructuras construidas en una parcela.
4. **EJES (Elementos lineales como calles y carreteras).** Son líneas en el mapa que representan elementos como calles, carreteras u otras estructuras lineales.
5. **ELEMLIN (Elementos cartográficos en forma de líneas).** Son características en el mapa representadas por líneas, como carreteras, ríos, etc.
6. **ELEMPUN (Elementos cartográficos en forma de puntos).** Son características en el mapa representadas por puntos, como ubicaciones específicas.

7. **ELEMTEX (Rótulos o etiquetas en el mapa).** Son las etiquetas o rótulos que proporcionan información adicional en el mapa, como nombres de lugares o detalles importantes.
8. **HOJAS (Divisiones de la cartografía urbana).** Son subdivisiones en la cartografía urbana que facilitan la organización y visualización de la información en áreas específicas.
9. **LÍMITES (Divisiones administrativas como municipios o zonas urbanas).** Se refiere a las divisiones administrativas que definen áreas geográficas específicas, como municipios o zonas urbanas.
10. **MAPA (Identificación de zonas con cartografía diferente).** Identifica áreas en el que la cartografía puede variar ligeramente, como por ejemplo entre áreas urbanas y rurales dentro de un mismo municipio.
11. **MASA (Agrupaciones de parcelas como manzanas urbanas o polígonos rurales).** Representa agrupaciones de parcelas, como manzanas en áreas urbanas y polígonos en áreas rurales.
12. **PARCELA (Parcelas catastrales).** Se refiere a divisiones específicas de la tierra con propósitos catastrales, utilizadas para la administración de la propiedad.
13. **SUBPARCE (Subparcelas que representan zonas específicas dentro de una parcela).** Son subdivisiones de parcelas que indican diferentes usos o cultivos en una misma parcela.

Esta información está agrupada por municipios de provincia en archivos comprimidos, lo que genera la necesidad de descompresión y unificación de todas las capas municipales en una única. Por ello, se crean 13 archivos en formato SHP acordes a estas tipologías, de manera que incluyen la información geográfica de toda comunidad autónoma extremeña. Ello se ha realizado mediante el archivo “Unir_shp_catastro.py” que utiliza librería `zipfile` para descomprimir los archivos y la librería *geo*`pandas` para generar el archivo unificado en formato SHP.

En primer lugar, se define el directorio principal que almacena los archivos ZIP con los Shapefiles (*directorio_principal*). Se utiliza la biblioteca `os` para iterar sobre los archivos ZIP en este directorio y crear un diccionario llamado *directorios*, que almacena los nombres de los archivos Shapefile en cada subdirectorio.

Luego, se filtran solo los subdirectorios que contienen archivos ZIP, excluyendo el directorio principal. Se crea un *directorio_temporal* donde se extraerán temporalmente los archivos ZIP.

El código itera sobre cada directorio con archivos ZIP, extrae cada archivo ZIP en una carpeta temporal, y crea una lista de todos los archivos extraídos (archivos Shapefile). Estos archivos Shapefile se identifican por su extensión ".SHP".

Se obtienen los nombres de archivo comunes en todos los archivos ZIP, y para cada nombre de archivo común, se crea una carpeta en el directorio principal. Los archivos extraídos que contienen ese nombre en su ruta se leen con geo`pandas`, se proyectan a EPSG:4326 (coordenadas latitud/longitud), y se combinan en un solo GeoDataFrame (*merged_gdf*).

Finalmente, se guarda el GeoDataFrame combinado en un nuevo archivo Shapefile en la carpeta recién creada en el directorio principal.

En resumen, el código automatiza el proceso de extracción, proyección y unión de archivos Shapefile contenidos en archivos ZIP, organizando y guardando los resultados en un formato consistente en el directorio principal.

Tras este preprocesado de los archivos SHP, se incorpora la información dada por los archivos ELEMLIN, MASA, PARCELA y CONSTU al proyecto en ArcGIS Pro Desktop.

3.2 Extracción de la cartografía vectorial municipal

A través del centro de descargas del Centro Nacional en Innovación Geográfica (CNIG) se extraen las tablas de información geográfica de los municipios de Extremadura. El archivo resultante de la descarga no requiere ningún pretratamiento previo para su lectura en ArcGIS.

Figura 9. Captura de pantalla la plataforma del CNIG de la cartografía de los municipios. [14]

Esta herramienta facilita la incorporación de toda la cartografía vectorial de los municipios de en un solo archivo SHP.

3.3 Extracción de los certificados de Extremadura

Para poder extraer esta información se requiere utilizar la técnica de rascado web o *web scrapping*. Esta técnica consiste en extraer información de sitios web mediante el uso de bots que simulan la navegación de la que podría hacer uso un usuario. Esta metodología accede a dicha información por medio de consultas a los elementos que conforman la web.

A continuación, se describen los archivos PY que contienen el proceso de extracción y tratamiento de esta información:

1. **0_WebScarppingExtremadura.py.** En este código Python se lleva a cabo el proceso de extracción de datos desde la página web oficial [9], mediante la técnica de rascado web. El objetivo es recuperar información relevante de certificados energéticos, abarcando un total de 45,520 registros. Los datos extraídos comprenden elementos relevantes tales como: la dirección, el código de referencia, la categoría de consumo, la categoría de emisiones de CO2 y la referencia catastral.

   El script comienza importando las librerías necesarias para llevar a cabo la automatización web y el procesamiento de datos. Se utilizan módulos específicos de `Selenium`, como `webdriver` y `Options`, así como "`ChromeDriverManager`" para gestionar el controlador de Chrome. Además, se importan las bibliotecas "`pandas`" y `time` para manipulación de datos y gestión del tiempo, respectivamente.

   A continuación, se formula la definición de una función llamada `tabla_pag()`. Esta función se encarga de extraer los datos almacenados en formato tabla dentro de la página web y devolverlos como un dataframe de `pandas`. Utiliza selectores CSS para localizar y organizar la información de la tabla.

   Seguidamente, se establece una variable *path* que representa la ruta donde se guardarán los archivos resultantes y se crean las listas de los municipios para las provincias de Badajoz y Cáceres. Esto último, se realiza por medio de la limpieza del objeto `<div id = "formulario:titutarMunicipio_panel"…` que contiene todos los municipios de cada provincia de Extremadura, extraída mediante la opción "Inspeccionar" en Google Chrome.

   Con los listados de provincias y municipios preparados, se procede a extraer la información de los registros de eficiencia energética de la página web. Para ello, se genera un bucle principal que itera sobre las provincias y, para cada provincia, sobre sus respectivos municipios. En cada iteración, se utiliza `selenium` para automatizar la interacción con la página web.

   Dentro del bucle, se crea una instancia del navegador Chrome y se navega a la página web correspondiente. Para establecer la conexión se inicia configurando las opciones del navegador,

estableciendo el modo `headless` para una ejecución sin interfaz gráfica visible. Luego, se inicializa el controlador de Chrome (`driver=webdriver.Chrome(ChromeDriverManager().install())`) y maximiza la ventana del navegador para garantizar una visualización completa de la página (`driver.maximize_window()`). A continuación, se navega a la URL específica de la página web con la función `driver.get([URL])`.

El código continúa interactuando con diversos elementos de la página mediante *selenium*. En primer lugar, realiza un clic en un elemento de clase `ui-menuitem-text`, lo que provoca la apertura del menú desplegable del campo "Provincia". Tras una pausa de un segundo para permitir la carga completa de la página, ingresa la provincia correspondiente en el formulario.

```
# Código

for i in range(len(provincias)):
    num_municipios = len(municipios[provincias[i]][:])

    for j in range(num_municipios):

        # nueva búsqueda

        options = Options()
        options.headless = True
        driver = webdriver.Chrome(ChromeDriverManager().install())
        driver.maximize_window()
        driver.get("https://asistenteagile.juntaex.es/AsistenteAGILE/AsistenteInfoCEEE.xhtml")

        driver.find_element(By.CLASS_NAME,"ui-menuitem-text").click()
        time.sleep(1)


        driver.find_element(By.ID,"formulario:titularProvincia_input").send_keys(provincias[i])
        time.sleep(1)
        selec_prov = '[data-item-label="' + provincias[i] + '"]'
        driver.find_element(By.CSS_SELECTOR, selec_prov).click()

        driver.find_element(By.ID,"formulario:titularMunicipio_input").send_keys(municipios[provincias[i]][j])
        time.sleep(1)
        selec_mun = '[data-item-label="' + municipios[provincias[i]][j] + '"]'
        driver.find_element(By.CSS_SELECTOR, selec_mun).click()


        driver.find_element(By.ID,"formulario:botonBuscar").click()
        time.sleep(1.5)
```

Figura 10. Conexión y búsqueda por municipios en Google Chrome

Posteriormente, se utiliza un selector CSS generado dinámicamente para hacer clic en el elemento que coincide con el municipio deseado dentro del campo "Municipio". Nuevamente, se introduce una pausa para asegurar la correcta ejecución de estas acciones.

Una vez seleccionados la provincia y el municipio, el código realiza un clic en el botón de búsqueda, identificado por el ID "formulario:botonBuscar". Este evento desencadena la búsqueda de certificados energéticos asociados al municipio y provincia previamente seleccionados. Este proceso de

interacción es crucial para obtener la información específica de certificados energéticos de cada municipio en Extremadura.

Después de realizar la búsqueda, el script inicia un bucle para extraer información de las tablas presentes en la página. La información se recopila en un dataframe de `pandas`. El bucle continúa avanzando a través de las páginas hasta que no hay más páginas disponibles.

Finalmente, para evitar posibles pérdidas de información y agilizar los tiempos de ejecución, los datos del dataframe se almacenan en formato Excel en el directorio indicado (*path*) con un nombre específico que sigue el siguiente formato: "Certificados_Extremadura_[Municipio].xlsx". Esta estructura facilita la localización de los municipios de los que no disponen de datos o bien no se han llegado a extraer durante el rascado de la web. De esta manera, si se produce algún error durante la extracción, es posible acceder directamente al municipio o provincia específica que generó dicho error, analizar el motivo de este y continuar extrayendo la información en el municipio en el que se produjo el problema.

Además, el código maneja situaciones excepcionales, como la falta de información en un municipio específico, y continúa con la siguiente iteración.

En general, este proceso resulta ser lento y complejo debido a los desafíos habituales del rascado de datos, como las posibles interrupciones de la conexión que pueden detener el código. Para abordar estas problemáticas, se implementan estrategias como la introducción de pausas estratégicas en la ejecución del código, especialmente después de realizar acciones que involucran la actualización de elementos en la página web. Estas pausas son esenciales para minimizar las interrupciones, ya que otorgan al sistema el tiempo necesario para cargar completamente la página antes de proceder con las siguientes acciones. Este enfoque contribuye a mejorar la estabilidad y eficiencia del proceso de extracción de datos, garantizando una ejecución más fluida y exitosa.

2. **1_Unificar_xlsx.py.** Este código en Python se unifica toda la información almacenada anteriormente en un único archivo, con los datos en bruto, bajo el nombre de "*Certificados_Extremadura_brutos.parquet*".

   En la primera parte del código se establece la ruta donde se encuentran estos archivos y, a continuación, se ejecuta un bucle que busca archivos con la extensión ".xlsx" en el directorio especificado. Los nombres completos de estos archivos se añaden a una lista llamada *list_archives*. Lista que servirá como referencia para acceder a cada archivo individual.

   El siguiente paso consiste en consolidar los datos de los archivos individuales en un único dataframe, denominado *df_certificados*. Se inicializa este dataframe como vacío y se itera sobre la lista de

archivos. En cada iteración, se lee el contenido del archivo Excel correspondiente y se agrega al dataframe principal. Se excluye la primera columna de cada archivo ya que, corresponde a los headers de las tablas y por tanto, no se debe considerar en el dataframe final.

Finalmente, el dataframe consolidado se guarda en dos formatos distintos. Primero, se genera un archivo Excel denominado "Certificados_Extremadura_brutos.xlsx", preservando así la estructura original de los datos. Luego, se almacena una versión más eficiente y compacta en formato PARQUET, bajo el nombre "Certificados_Extremadura_brutos.parquet".

2_Limpieza_certificados.py. Este código en Python desempeña un papel crucial al realizar una exhaustiva limpieza, normalización y búsqueda de registros con valores de referencia catastral erróneos en un conjunto de datos. Para llevar a cabo estas tareas, se emplean bibliotecas esenciales como `re` y `unicededata`, permitiendo acciones precisas de normalización de direcciones. A través de técnicas de limpieza y normalización de datos, utilizando estructuras de reemplazo de datos mediante diccionarios y extracción de información específica de cadenas de texto, se ajustan las columnas “Dirección”, “Referencia Catastral” y “Emisiones de CO2” para que sean compatibles con los registros de la base de datos catastral, incluyendo información alfanumérica.

La normalización y limpieza de las columnas "Dirección" y "Referencia Catastral" se presentan como desafíos, dado que muchos registros contienen múltiples referencias catastrales, y varias de ellas carecen de la codificación catastral adecuada. Como resultado, se extraen datos limpios, conservando únicamente las referencias catastrales válidas, es decir, aquellas compuestas por 14 o 20 dígitos. El resultado de este proceso se almacena en el archivo "Data_CE_provisional.parquet".

Con el objetivo de no perder información relacionada con referencias catastrales inadecuadas o poco ortodoxas, se realiza una búsqueda adicional de estas referencias en los datos catastrales de vivienda almacenados en "Data_Catastro.parquet". Se identifican las direcciones que carecen de valores de referencia catastral adecuados, almacenándolos en la variable *df_norm*. Luego, se desarrolla un código capaz de extraer información detallada de la columna "Dirección", incluyendo la provincia, el tipo de vía pública, el nombre de la vía, el primer número de póliza con su letra, el piso y la puerta. La complejidad de las posibles combinaciones para esta extracción motiva el uso de elementos de programación, como definiciones, cuyo entendimiento se recomienda explorar directamente en el código.

Concluyendo, el resultado final es un conjunto de datos limpio y adecuado, compuesto por 37,240 registros, y almacenado en formatos XLSX y PARQUET con el nombre *Data_CE*. Este archivo representa una versión depurada y mejorada del conjunto de datos original, listo para análisis y aplicaciones posteriores.

A continuación, se realiza una descripción más detallada del código.

El script comienza importando las bibliotecas de *pandas*, *numpy*, *unicodedata*, *re* y *warnings*.

Seguidamente, define varias funciones con propósitos específicos:

- Función `elimina_diacriticos` se encarga de quitar los diacríticos de una cadena de texto, preservando la letra "ñ". Utiliza las bibliotecas *re* y *unicodedata* para normalizar los caracteres.
- Función `cont_palabras` opera sobre una lista de cadenas de texto, contando la frecuencia de cada palabra y presentando una lista ordenada, así como los 50 términos más frecuentes. Esto se logra mediante el uso de diccionarios para el conteo de palabras y sus frecuencias.
- Función `describe_columnas`, realiza una descripción detallada de las características de las columnas especificadas de un dataframe. Presenta información como el número de filas, el recuento de valores únicos y sus frecuencias. Utiliza diccionarios para llevar a cabo el conteo de frecuencias y presenta la información de manera organizada.
- Función `extrae_nom_1_pol_nom_1_letra_planta_puerta` tiene como objetivo extraer información específica, como el número de póliza, la letra, la planta y la puerta, de una cadena de texto que representa una dirección. Realiza una limpieza previa y utiliza expresiones regulares para identificar y extraer la información pertinente.

Además de estas funciones, el código realiza operaciones iniciales, como la definición de rutas de archivo para conjuntos de datos de certificados y datos catastrales. Lee el conjunto de datos de certificados desde el archivo Excel "Certificados_Extremadura_brutos.xlsx", proporcionando estadísticas iniciales sobre el dataframe, como el número de filas y columnas, así como recuentos de valores nulos en columnas específicas. También elimina duplicados del dataframe y selecciona solo las columnas "Dirección", "Emisiones de CO2" y "Referencia Catastral". Esto da lugar al dataframe *df_cert*.

Seguidamente se procede a normalizar la columna "Emisiones de CO2" de *df_cert*. Comienza con un análisis inicial de la distribución de los datos en esta columna utilizando la función `describe_columnas`. Este análisis proporciona información sobre el número de filas, los valores únicos y sus frecuencias en la columna "Emisiones de CO2", lo que es crucial para comprender la naturaleza de los datos.

A continuación, se manejan de datos faltantes o incorrectos, reemplazando los valores "?" en la columna "Emisiones de CO2" con valores nulos (`None`), y luego se eliminan las filas que contienen valores nulos en esta columna.

Tras la normalización de la referencia catastral, se desarrolla la parte más compleja del código, bautizada como "PASO 2 - NORMALIZAR COLUMNA DIRECCIÓN". Su objetivo principal es mejorar la calidad y uniformidad de la información contenida en la columna “Dirección”, facilitando así su manejo y análisis. Se utilizará la información de la columna “Dirección” para corregir los registros de la columna que concierne al código de referencia catastral de cada registro (“ref_cat”).

En primera instancia, se carga el dataframe con la información catastral procedente del archivo “Data_Catastro.parquet”, obtenido previamente en la sección 3.1.1 (*df_cat*). Se introducen transformaciones en las columnas “nom_mun” y “nom_prov” de este dataframe, eliminando diacríticos y convirtiendo todos los caracteres a mayúsculas. Estos pasos son fundamentales para asegurar la coherencia y uniformidad en la información.

A continuación, se implementa la normalización de direcciones en *df_cert*. Esto implica la eliminación de diacríticos, la conversión a mayúsculas y la aplicación de reglas específicas definidas en un diccionario. Además, se filtran aquellas direcciones que no contienen tipos de vía válidos, contribuyendo así a la limpieza y homogeneización del conjunto de datos.

```
# Normalización de las direcciones
for key, value in dic_normalizar.items():
    df_cert['dir_norm'] = df_cert['dir_norm'].str.replace(key, value)

df_cert['dir_norm'] = df_cert['dir_norm'].apply(lambda i: i.replace('C ', 'CL') if i[:2] == 'C ' else i)
df_cert['dir_norm'] = df_cert['dir_norm'].apply(lambda i: i.replace('PA ', 'PS') if i[:3] == 'PA ' else i)
df_cert['dir_norm'] = df_cert['dir_norm'].apply(lambda i: re.sub(r'(\b[A-Z]+\b)\s+(\d)', r'\1, \2', i))
df_cert['dir_norm'] = df_cert['dir_norm'].apply(lambda i: re.sub(r'(\d)0([A-Z])', r'\1\2', i))

list_dir_sin_norm = [i for i in df_cert['dir_norm'] if i[:2] not in list_vp_sig]
list_dir_norm = [i for i in df_cert['dir_norm'] if i[:2] in list_vp_sig]

# Creación df_cert sin valores nulos en ref_cat
df_cert_con_ref_cat = df_cert[~df_cert['ref_cat'].isnull()]
df_cert.reset_index(drop=True, inplace=True)

# Aseguramos el tipo str de ref_cat
df_cert['ref_cat'] = df_cert['ref_cat'].astype(str)

# Eliminar los valores no alfanuméricos de los registros de ref_cat y separar los registros
df_cert_con_ref_cat['ref_cat'] = df_cert_con_ref_cat['ref_cat'].replace(r'\W', ' ', regex=True)
df_cert_con_ref_cat['ref_cat'] = df_cert_con_ref_cat['ref_cat'].replace(' Y ', ' ', regex=True)
df_cert_con_ref_cat['ref_cat'] = df_cert_con_ref_cat['ref_cat'].apply(lambda x: x.split())

new_rows = []
# cat_ant = ''
for idx, row in df_cert_con_ref_cat.iterrows():
    ref_cat = row['ref_cat']
    for cat in ref_cat:
        new_row = row.copy()
        new_row['ref_cat'] = cat
        new_rows.append(new_row)


df_cert_con_ref_cat = pd.DataFrame(new_rows)
df_cert_con_ref_cat = df_cert_con_ref_cat.drop_duplicates(subset = ['ref_cat'])

df_cert_con_ref_cat['len_ref_cat'] = df_cert_con_ref_cat['ref_cat'].apply(lambda x: len(str(x)))
df_cert_con_ref_cat.reset_index(drop=True, inplace=True)
describe_columnas(df_cert_con_ref_cat,['len_ref_cat'])

# Definición del dataframe a normalizar para extraer el valor real de la referencia catastral
df_cert_20 = df_cert_con_ref_cat[df_cert_con_ref_cat['len_ref_cat'] == 20][['dir_norm','ref_cat','Emisiones de CO2']]
df_cert_14 = df_cert_con_ref_cat[df_cert_con_ref_cat['len_ref_cat'] == 14][['dir_norm','ref_cat','Emisiones de CO2']]
df_cert_20.reset_index(drop=True, inplace=True)

# En caso de tener referencia catastral == 14
df_cat['ref_cat_14'] = df_cat['ref_cat'].apply(lambda x: x[:14])
df_cert_14 = df_cert_14.rename(columns = {'ref_cat':'ref_cat_14'}).merge(df_cat[['ref_cat_14','ref_cat']], on='ref_cat_14', how='inner')
df_cert_14 = df_cert_14.drop('ref_cat_14', axis = 1).drop_duplicates(subset = ['ref_cat'])
df_cert_14.reset_index(drop=True, inplace=True)

# Aumentar el número de registros con certificados de longitud 20
df_cert_20 = pd.concat([df_cert_20, df_cert_14], ignore_index=True)
```

Figura 11. Normalización de direcciones de los certificados energéticos

El siguiente paso es la normalización de referencias catastrales (columna “ref_cat”) del dataframe *df_cert_20*. Este dataframe contiene registros que poseen referencias catastrales de longitud 20 y 14 caracteres, junto con información sobre la dirección normalizada y las emisiones de CO2. A continuación, se detalla cada paso de forma detallada.

Primero, se crean dos subconjuntos de datos, *df_cert_20* y *df_cert_14*, que contienen registros con referencias catastrales de longitud 20 y 14, respectivamente. Los datos en *df_cert_20* se reinician para mejorar la manipulación de índices.

Luego, para las referencias catastrales de longitud 14, se realiza un proceso de coincidencia con el dataframe auxiliar *df_cat*. Se crea una nueva columna “ref_cat_14” en *df_cat*, que almacena los primeros 14 caracteres de “ref_cat”. Después, se renombra la columna “ref_cat” en *df_cert_14* a “ref_cat_14” y se fusiona con *df_cat*. Se eliminan duplicados basados en “ref_cat” y se reinician los índices.

A continuación, se aumenta el número de registros en *df_cert_20* al concatenar los datos de *df_cert_14*, y se genera un nuevo dataframe denominado *df_norm*, que contiene registros pendientes de normalización. Estos registros corresponden a aquellos con referencias catastrales de longitudes distintas de 20 o 14 y aquellos cuya referencia catastral es “nan”.

Si hay duplicados en la columna “ref_cat” del dataframe *df_cert_20*, se selecciona la fila con la valoración más alta de “Emisiones de CO2” y se eliminan las duplicadas. Se reinician los índices de los dataframes *df_norm* y *df_cert_20*, y se introducen columnas adicionales “No tenía ref_cat válida” en *df_norm* y *df_cert_20*, ambas inicializadas en 0. Esta última columna se crea con la finalidad de marcar aquellos registros en los cuales se haya renombrado la referencia catastral.

A continuación, se inicia un bucle para corregir la referencia catastral de los registros de certificados energéticos de Extremadura. Para ello, el bucle comienza inicializando una lista llamada *new_rows*, que se utilizará para almacenar las filas que contengan referencias catastrales encontradas durante la búsqueda. Luego, itera sobre cada fila del dataframe *df_norm*, donde *i* representa la dirección normalizada y *j* es la referencia catastral de la fila actual.

Dentro del bucle, se realiza una búsqueda de las provincias presentes en la dirección normalizada mediante un bucle `for`. Se almacenan las posiciones de inicio y fin de las ocurrencias de las provincias en la lista *posicion_prov*. Además, se determina la posición del municipio en la dirección normalizada y se valida la existencia de la provincia.

Posteriormente, se verifica que la dirección normalizada no se encuentre dentro de la lista de direcciones sin normalizar (*list_dir_sin_norm*). En este caso se crean las variables relacionadas con la

dirección normalizada, como el tipo de vía pública (“nom_vp_sig”), el municipio (“nom_mun”), y la provincia (nom_prov).

A continuación, se elige entre dos opciones para buscar el nombre de la vía pública en el dataframe *df_cat*. La opción A utiliza filtros booleanos, mientras que la opción B utiliza la función `loc`. Se comprueba que utilizar la función `loc` mejora la eficiencia del código, por lo que se opta por esta opción. Se busca el nombre de la vía pública en la dirección normalizada, manejándose casos de múltiples coincidencias, es decir se creará una lista con posibles nombres de vía pública válidos. Por lo tanto, en función de la longitud de la lista se actuará de una manera concreta.

En el caso de encontrar un único nombre de vía pública válido (con una longitud igual a 1), se procede a reformular la referencia catastral. Esta reformulación implica la separación de los números y letras de la referencia catastral original, seguida por el cálculo de una configuración específica denominada “ref_config_ref_cat”. La razón detrás de este cálculo radica en los errores tipográficos presentes en las referencias catastrales del dataframe *df_cert*. En numerosas ocasiones, estas referencias presentan algún error en las diferentes partes que componen la referencia catastral, como puede ser el código de parcela, el número de cargo o código de control, lo que da como resultado una longitud de la referencia catastral distinta de 20 caracteres. Al determinar la configuración de este número de referencia catastral, es posible identificar la ubicación exacta del error tipográfico. De esta manera, se utiliza la parte válida del código para llevar a cabo la búsqueda de la referencia catastral real en el dataframe *df_cat*, teniendo en cuenta ciertas condiciones.

```
# Reformular referencia catastral
# Separar los números de las nom_1_letras y calcular la longitud de cada valor
config_ref_cat = [len(elem) for elem in re.findall(r'\d+|[A-Za-z]+', j)]
sep_ref_cat = [elem for elem in re.findall(r'\d+|[A-Za-z]+', j)]

ref_config_ref_cat = [ref if len(ref) == len(config_ref_cat) else list_ref_config_ref_cat[0] for ref in list_ref_config_ref_cat][0]

if len(config_ref_cat) == len(ref_config_ref_cat):
    mantisa_config_ref_cat = [1 if val == ref else 0 for val, ref in zip(config_ref_cat, ref_config_ref_cat)]

    pos = 0
    re_ref_cat = ''
    while pos < len(ref_config_ref_cat):
        if mantisa_config_ref_cat[pos] == 1:
            re_ref_cat += sep_ref_cat[pos]
        else:
            re_ref_cat += '.*'
        pos += 1
```

Figura 12. Formulación de la configuración y de la parte válida de la referencia catastral

Acto seguido, se lleva a cabo la extracción de la referencia catastral, considerando diversas situaciones según las características de la dirección y la presencia de ciertos términos clave como "PARCELA", "PARCELAS" o "TRASERA". Esto da lugar a tres ramas condicionales:

a) **Extracción de referencia catastral con** "**re_ref_cat**"**.** En la primera rama condicional, se verifica la presencia de un único nombre de vía pública válido y se evalúa si la dirección no contiene términos clave como "PARCELA", "PARCELAS" o "TRASERA", o si contiene alguno de ellos. En caso de contener dichos términos clave, se buscan expresiones numéricas en el texto de la dirección y se itera sobre los resultados para extraer detalles como el número de policía, la letra, la planta y la puerta. Por otro lado, si la dirección no contiene términos clave, se procede directamente a separar los elementos relevantes del texto, como el número de policía, la letra, la planta y la puerta. Posteriormente, se convierte el número de póliza a un valor entero y se realiza la búsqueda de la referencia catastral en *df_cat* utilizando los parámetros conocidos de estas nuevas variables creadas ("re_ref_cat", "nom_vp", "num_1_pol", "nom_prov" y "nom_mun"). Si se encuentra un único registro de referencia catastral, se agrega la fila actualizada al dataframe resultante *new_rows*.

b) **Manejo de referencias catastrales poco ortodoxas o desconocidas.** En la segunda rama condicional, se aborda la situación en la que no se encuentra una referencia catastral válida o esta es poco ortodoxa ("re_ref_cat"). Se aplican acciones similares a las condiciones tanto con términos clave como sin ellos, teniendo en cuenta detalles adicionales como las siglas del tipo de vía pública. Si se identifica una única referencia catastral válida, se agrega la fila actualizada al dataframe *new_rows* y se etiqueta la entrada con "No tenía ref_cat válida". Este enfoque permite gestionar eficientemente referencias catastrales inusuales o desconocidas durante el proceso de extracción y validación.

Finalmente, se genera un dataframe denominado *df_cert_definitivo* que consolida tanto los nuevos registros obtenidos (*new_rows*) como aquellos de 20 caracteres (*df_cert_20*). Se lleva a cabo la eliminación de duplicados basados en la columna "ref_cat" y se restablecen los índices para una estructura ordenada. Además, se incorpora una columna adicional, "len_ref_cat", que refleja la longitud de la referencia catastral.

Con el propósito de validar y revisar los registros de este dataframe definitivo, se emplea la función `describe_columnas`, proporcionando así estadísticas descriptivas específicamente para la columna "len_ref_cat" en *df_cert_definitivo*.

En la etapa final del proceso, los resultados finales se exportan en formatos de archivo PARQUET y EXCEL, almacenados en las rutas predefinidas (`path_CE + "Data_CE.parquet" y path_CE + "Data_CE.xlsx"`).

3.4 Extracción de la renta media por hogar

Entre las publicaciones del Instituto Nacional de Estadística se encuentran los datos de renta media anual por hogar de las provincias de Badajoz y Cáceres [15]. Estos se pueden extraer en formato CSV donde la renta media viene desagregada a un nivel de distrito.

Tras su extracción se unifican los CSV, se extrae la renta neta media por hogar del 2020 para cada distrito y se crea un nuevo archivo Excel con los datos de unificados y tratados para su posterior incorporación a en el conjunto de datos final descrito en la sección 3.6.1

El código responsable de estas operaciones se denomina "0_Unir_Limpiar_Extraer_Renta.py"**.** El proceso comienza con la configuración inicial del entorno de trabajo en Python mediante la importación de las bibliotecas necesarias, como `pandas`, `os` y `warnings`. Se establece que las opciones de visualización permitan mostrar todas las columnas, y se configura la supresión de advertencias para mejorar la legibilidad del código, tal y como veíamos en secciones anteriores.

A continuación, se definen funciones esenciales, como `describe_columnas`, destinada a proporcionar estadísticas descriptivas detalladas sobre las columnas de un dataframe, incluyendo el número de filas y la frecuencia de valores únicos.

Posteriormente, se establece la ruta del directorio que contiene los archivos CSV descargados de Catastro. Se procede a la unificación de múltiples archivos CSV, presentes en el directorio especificado, en un solo dataframe llamado *df_renta*. Durante este proceso, se eliminan duplicados para garantizar la coherencia y fiabilidad de los datos.

Seguidamente se da comienzo a la fase de limpieza y normalización, centrando la atención en la "Renta neta media por hogar" del año 2020 para cada distrito. Se realiza la eliminación de espacios en blanco al final de las columnas de tipo objeto y se filtran los datos relevantes, excluyendo subdivisiones ("Secciones" nulas). Además, se lleva a cabo un procesamiento adicional en la columna "Total", convirtiendo el carácter "." a valores nulos (`None`), y se eliminan las filas correspondientes.

A continuación, se extrae el código INE de la columna "Distritos", utilizando información de las columnas "Municipios". Este paso es crucial para la identificación y clasificación adecuada de los datos.

Finalmente, el dataframe resultante se almacena en un archivo Excel denominado "Renta neta media por hogar_2020.xlsx". Esta exportación se realiza en la misma ubicación que los archivos CSV originales.

3.5 Extracción de la zona climática

La obtención de la información sobre las zonas climáticas de los municipios de Extremadura se realiza considerando el Código Técnico de la Edificación, específicamente en el Documento Básico de Ahorro de Energía [16]. Este documento establece las zonas climáticas según la provincia y la altitud relativa al nivel del mar de cada localidad.

Para llevar a cabo este proceso, se recopila previamente la altura de los municipios de Extremadura, dato crucial para la correcta asignación de la zona climática correspondiente. Posteriormente, se organiza esta información en un archivo Excel, proporcionando una referencia visual y estructurada de las zonas climáticas asociadas a cada municipio.

Este enfoque permite incorporar un elemento clave en la clasificación climática de los municipios, contribuyendo así a la comprensión detallada de las condiciones ambientales específicas de cada localidad. La creación del archivo Excel no solo facilita la gestión de estos datos, sino que también brinda la posibilidad de realizar análisis y consultas de manera eficiente, consolidando así la información climática esencial para el ámbito de la edificación y el ahorro energético en la región de Extremadura.

3.6 Datasets

Los datasets o conjunto de datos utilizados en este proyecto se dividen en dos grupos: los datasets con la información de la vivienda y los datasets con la información cartográfica.

A continuación, se describen como se conforman los datasets utilizados para dar lugar al resultado final del proyecto.

3.6.1 Datasets de vivienda

Una vez descargados los datos catastrales, los certificados energéticos, la renta media por hogar y la zona climática, se generan tres conjuntos de datos distintos:

- **Dataset_Extremadura.** Este conjunto de datos combina la información de las cuatro fuentes mencionadas. Incluye un total de 959,451 registros de viviendas, cada uno caracterizado por 13 variables numéricas y 19 variables categóricas o alfanuméricas. El objetivo fundamental de crear este conjunto de datos es poder incorporar esta información en los mapas de ArcGIS.
- **Dataset_model.** En este conjunto de datos se procesa la información del “Dataset_Extremadura”. Se lleva a cabo una transformación de algunas variables categóricas en variables binarias. Las variables afectadas por esta transformación son: clase del inmueble, tipo de reforma, provincia, zona climática, uso de inmueble y calificación energética. Además, se eliminan variables numéricas como el número de orden de construcción, número de polígono, código postal o área de la finca en metros cuadrados. Las variables restantes no numéricas son también excluidas. Esto resulta en un conjunto de datos

conformado únicamente por variables numéricas, adecuadas para ser procesadas por un modelo de Machine Learning (ML) diseñado para predecir la certificación energética. En este proceso de filtrado, se excluyen todos los registros que no cuentan con certificación energética. El *dataset_model* cuenta con un total de 42,660 registros, descritos por 43 variables numéricas.

- **Dataset_predict.** Las características descriptivas de cada registro en este conjunto de datos son las mismas que las del *dataset_model*. Sin embargo, este conjunto solo incluye registros de aquellos para los cuales no se conoce la calificación energética. El propósito de este conjunto de datos es permitir la predicción de la certificación energética para estos registros específicos.

Estos tres conjuntos de datos se obtienen mediante el uso del código en Python denominado "0_Creacion_Dataset.py". Este genera los tres archivos en formato PARQUET.

El código Python hace uso de varias bibliotecas esenciales para el manejo de datos y análisis geoespacial. Se importan las librerías: "`pandas`" para la manipulación de dataframes, `re` para expresiones regulares, geo`pandas` y `shapely.geometry` para trabajar con datos geoespaciales, y `unicodedata` para normalizar caracteres y eliminar diacríticos.

Entre las definiciones destacadas, se encuentran `elimina_diacriticos`, que elimina los diacríticos de un texto, y `describe_columnas`.

A continuación, se realiza la lectura de los archivos PARQUET y EXCEL con la información alfanumérica de los datos catastrales (*df_cat*), los certificados energéticos (*df_CE*), las zonas climáticas de cada municipio (*df_ZC*) y la renta neta media por hogar (*df_renta*) provenientes de "Data_Catastro.parquet", "Data_CE.parquet", "zonaClimatica_municipios.xlsx", y "Renta neta media por hogar_2020.xlsx", respectivamente.

La unificación de datos sigue, con la inclusión de la información de altitudes y zonas climáticas al dataframe *df_cat*. En primer lugar, sobre *df_ZC*, se calcula la longitud del código INE para generar los códigos de provincia ("cod_prov_INE") y municipio ("cod_mun_INE") del INE ya que, es a partir de estos códigos a través de los cuales se fusiona la información de las zonas climáticas con la información catastral. Además, las columnas resultantes se renombran para mayor claridad.

A continuación, se incorporan los datos relacionados con la renta neta media por hogar en el año 2020 ("renta_neta_por_hogar") desde el dataframe *df_renta* al dataframe principal *df_cat*. Se realiza un procesamiento adicional para llenar los valores nulos de esta variable con la media de las rentas en el municipio cuando sea necesario.

Luego, se anexan los certificados energéticos desde el dataframe *df_CE* al dataframe *df_cat* y se gestiona la posible duplicidad de registros basándose en la columna "ref_cat".

Posteriormente, se eliminan columnas innecesarias del dataframe resultante (*dataset*), y se realiza una transformación de códigos de clase de inmueble para una mayor claridad en la lectura de los datos.

Finalmente, se exporta el dataframe resultante como un archivo parquet ("Dataset_Extremadura.parquet") y como un archivo shapefile ("Dataset_Extremadura.shp") para su uso en el sistema de información geográfica (SIG) ArcGIS. Esta parte del código es conveniente destacarla ya que, requirió un estudio profundo de la librería de geo`pandas`.

```
# Extraemos el resultado final en archivo shapefile para posteriormente importarlo en ArcGIS

dataset.columns = dataset.columns.map(elimina_diacriticos)
geometry = [Point(xyz) for xyz in zip(dataset['longitud'], dataset['latitud'], dataset['altitud'])]
gdf = gpd.GeoDataFrame(dataset, geometry=geometry)
# Guarda el GeoDataFrame como un shapefile
gdf.to_file(path_resultado + 'Dataset_Extremadura.shp', driver='ESRI Shapefile')
```

Figura 13. Transformación de un dataframe a un shapefile

La primera línea de código renombra las columnas del dataframe *dataset* aplicando la función `elimina_diacriticos` a cada nombre de columna, eliminando así, los diacríticos (acentos, virgulillas, etc.) de las letras en los nombres de las columnas, lo que evita problemas con la codificación de caracteres posterior.

A continuación, se introduce una nueva columna denominada "geometry" en el dataframe original. Esta columna contiene objetos tipo Point de la biblioteca `shapely.geometry`, construidos a partir de las columnas "longitud", "latitud", y "altitud". Dichos objetos representan las coordenadas espaciales de cada registro como un punto tridimensional.

Posteriormente, se crea un GeoDataFrame llamado *gdf* mediante la combinación del dataframe original y la columna "geometry". El GeoDataFrame se utiliza para manejar los datos geoespaciales de las coordenadas y se aprovecha las funcionalidades de las bibliotecas geoespaciales, como `GeoPandas`.

Finalmente, el GeoDataFrame *gdf* se exporta como un archivo SHP en la ruta especificada. La opción `driver="ESRI Shapefile"` define el formato del archivo, optando en este caso por el formato Shapefile "ESRI", siendo "ESRI" la empresa que motoriza ARGIS.

Tras obtener el primer archivo utilizado para la representación geográfica de las viviendas en ArcGIS, se procede a preparar los datos la aplicación de estos en modelos de ML. En primer lugar, se realiza un análisis para identificar valores nulos y ceros en distintas variables del conjunto de datos, proporcionando una visión

general de la calidad de los datos. Posteriormente, se procede a la eliminación de variables categóricas consideradas poco relevantes para el modelo.

Además, se efectúa una depuración más profunda al descartar registros que carecen de coordenadas geoespaciales, siendo estas críticas para el posterior mapeo en un Sistema de Información Geográfica (GIS). Se eliminan también algunas columnas numéricas que se consideran menos relevantes para el análisis.

Notesé que, únicamente se pueden aplicar a un modelo de ML variables numéricas. Por ello, una parte crucial de la preparación de datos implica la transformación de variables categóricas mediante la creación de variables ficticias utilizando el método de codificación "one-hot" mediante la función de "`pandas`" "get_dummies()". Esto se aplica a variables como el tipo de inmueble, reformas realizadas, provincia, zona climática, uso del inmueble y emisiones de CO2. Se recuerda que esta última variable es la que contiene la calificación energética de la vivienda.

Posteriormente, se lleva a cabo una partición del conjunto de registros en dos entidades específicas: *dataset_model* y *dataset_predict*, basándose en la presencia de certificados energéticos identificados por las letras "A" a "G". El primer dataframe, *dataset_model*, engloba todos los registros que contienen información detallada sobre certificación energética. En contraste, *dataset_predict* alberga aquellos registros que carecen de esta información.

Esta división se realiza con la finalidad de utilizar los datos en *dataset_model* para la experimentación y validación de diversos modelos de aprendizaje automático. Por otro lado, *dataset_predict* se reserva para asignar calificaciones energéticas a las viviendas que carecen de esta clasificación, en caso de que los resultados obtenidos con *dataset_model* sean positivos.

Finalmente, los conjuntos de datos resultantes se exportan en formato PARQUET y EXCEL, con el propósito de utilizarlos en la fase de análisis realizada en JupyterLab.

Por último, en relación con el archivo generado "Dataset_Extremadura**"**, se han detectado inconvenientes durante su lectura en ArcGIS Pro Desktop que requieren atención. Tras una evaluación detallada de los puntos en ArcGIS junto con las capas mencionadas en la sección 3.6.2, se ha observado que algunos de estos puntos no se encuentran dentro de las capas correspondientes a ciertas parcelas. Además, se ha notado que la mayoría de estos puntos carecen de la calificación energética, generando así una carga de información poco relevante en el mapa.

Como solución a este problema, se ha generado un nuevo archivo SHP desde ArcGIS, titulado "Dataset_Extremadura_Certificados_Existentes", que contiene exclusivamente los registros que poseen certificación energética. Esta medida ayuda a focalizar la representación gráfica en los puntos relevantes, mejorando la claridad e información del mapa.

Se ha identificado otro inconveniente en el mismo documento, relacionado con la ilegibilidad en ArcGIS al intentar utilizarlo como capa de puntos. Este problema surge debido a un error de interpretación del sistema de coordenadas del shapefile. Para resolver esta dificultad, se ha optado por eliminar el archivo SHP del conjunto de archivos que conforman el shapefile. Esta acción permite leer los datos en su formato bruto (archivo DBF), presentados en una tabla. Posteriormente, se asigna un sistema de coordenadas adecuado a esta tabla, facilitando así la representación gráfica de la capa de puntos.

Este enfoque integral ha demostrado ser efectivo para abordar ambos problemas, resultando en un mapa más claro y eficiente en ArcGIS Pro Desktop.

3.6.2 Capas cartográficas

Las capas cartográficas no son más que un conjunto de datos diseñado específicamente para representar datos geográficos.

A continuación, se describe la información cartográfica vectorial, previamente presentadas en las secciones 3.1.2 y 3.2, que conforman las diferentes capas del mapa del proyecto:

- **Municipios_Extremadura.** Se trata de una capa poligonal que contiene información del nombre del municipio, la cantidad de registros de vivienda y la moda de las certificaciones energéticas que contiene, su área y su longitud. Esta capa se crea a partir de la cartografía vectorial municipal cuya obtención se describe en la sección 3.2.
- **MASA.** Se trata de una capa poligonal que contiene información de la referencia catastral, la cantidad de registros de vivienda y la moda de las certificaciones energéticas que contiene, la delegación, el número de municipio, las coordenadas cartográficas, su superficie y su fecha de alta. Esta capa se crea a partir de la cartografía vectorial de catastro cuya obtención se describe en la sección 3.1.2.
- **PARCELA.** Se trata de una capa poligonal que contiene información de la referencia catastral, la cantidad de registros de vivienda y la moda de las certificaciones energéticas que contiene, el número de municipio, las coordenadas cartográficas, su superficie y su fecha de alta. Esta capa se crea a partir de la cartografía vectorial de catastro cuya obtención se describe en la sección 3.1.2.
- **ELEMLIN.** Se trata de una capa de elementos lineales que representan y remarcan elementos como carreteras, ríos, etc. Esta capa se crea a partir de la cartografía vectorial de catastro cuya obtención se describe en la sección 3.1.2.
- **CONSTRU.** Se trata de una capa poligonal que contiene información de los elementos constructivos de las parcelas catastrales. Sin embargo, esta capa se encuentra oculta dado que no favorece la interacción del usuario con el mapa. Esta capa se crea a partir de la cartografía vectorial de catastro cuya obtención se describe en la sección 3.1.2.

Es importante destacar que la cantidad de registros de vivienda y la moda de las certificaciones energéticas que se calcula para cada capa, se realiza por medio de las definiciones escritas en el lenguaje de Python, ya que ArcGIS Pro Desktop cuenta con una interfaz de Python incorporada. Además, para poder aplicar estas definiciones es necesario aplicar el modelo de superposición definido en la sección 3.7.2.

En las figuras que se muestran a continuación, de describe el código utilizado para incorporar esta información a cada capa. Nótese, que el usuario deberá modificar las variables de entrada en función de la capa a la que desee incorporar la moda y el recuento de registros de vivienda.

```python
import arcpy
from collections import defaultdict, Counter

# Definir variables de entrada
capa_input = "Dataset_Extremad_SpatialJoin_PARCELA"
capa_output = "PARCELA"
var_common = "REFCAT"

# Crear un diccionario para almacenar las emisiones por var_common
emi_dict = defaultdict(list)

# Abrir un cursor de búsqueda en la capa_input
with arcpy.da.SearchCursor(capa_input, [var_common, "Emisiones"]) as cursor:
    for row in cursor:
        id_parcela = row[0]
        emisiones = row[1]
        emi_dict[id_parcela].extend(emisiones)  # Extender la lista en lugar de agregar

# Ordenar cada lista de letras en orden descendente en moda_dict con el fin de escoger la mejor CE en caso de igualdad de modas
for key, value in emi_dict.items():
    emi_dict[key] = sorted(value, reverse=False)

# Crear un diccionario para mapear valores  con resultados de moda
moda_dict = {}

# Calcular la moda para cada id_parcela y mapear los resultados
for id_parcela, emisiones_lista in emi_dict.items():
    if emisiones_lista:
        moda = Counter(emisiones_lista).most_common(1)[0][0]
        moda_dict[id_parcela] = moda

# Agregar una nueva columna "Moda_CE" a la capa MASA
arcpy.AddField_management(capa_output, "Moda_CE", "TEXT")

# Actualizar la capa MASA con los valores de moda
with arcpy.da.UpdateCursor(capa_output, [var_common, "Moda_CE"]) as cursor:
    for row in cursor:
        moda = row[0]
        if moda in moda_dict:
            row[1] = moda_dict[moda]
            cursor.updateRow(row)
```

Figura 14. Incorporación de la moda de certificados energéticos en la capa PARCELA

```python
import arcpy
from collections import defaultdict

# Definir la capa de parcelas y la capa PARCELA
capa_input = "Dataset_Extremad_SpatialJoin_MUNICIPIO"
capa_output = "Municipios_Extremadura"
var_common = "NAMEUNIT"

# Crear un diccionario para almacenar la cantidad de registros por ID de parcela
registros_dict = defaultdict(int)

# Abrir un cursor de búsqueda en la capa de parcelas
with arcpy.da.SearchCursor(capa_input, [var_common]) as cursor:
    for row in cursor:
        id_parcela = row[0]
        registros_dict[id_parcela] += 1  # Contar registros por ID de parcela

# Agregar una nueva columna "Num_Registros" a la capa PARCELA
arcpy.AddField_management(capa_output, "Num_Reg", "TEXT")

# Actualizar la capa PARCELA con la cantidad de registros por cada ID de parcela
with arcpy.da.UpdateCursor(capa_output, [var_common, "Num_Reg"]) as cursor:
    for row in cursor:
        id_parcela = row[0]
        if id_parcela in registros_dict:
            row[1] = str(registros_dict[id_parcela])
            cursor.updateRow(row)
```

Figura 15. Incorporación del número de registros de vivienda en la capa MUNICIPIO

3.7 Modelos

Los modelos planteados en este proyecto se definen acorde a las dos variantes de análisis planteadas como objetivos de este trabajo: la predicción de la certificación energética de las viviendas y la detección de zonas de alto potencial de renovación energética.

3.7.1 Modelo predictivo de certificación energética

Con el objetivo de predecir la certificación energética de una vivienda, se analizan diferentes modelos de ML para determinar cuál de ellos es el más adecuado para predecir un tipo de certificación. Cada modelo está diseñado para predecir exclusivamente la probabilidad de que el registro corresponda a un tipo particular de certificación. El valor del certificado energético atribuido al registro será aquel proveniente del modelo cuya probabilidad de predicción sea más alta. Por ejemplo, si un modelo predice con un 98% de probabilidad la certificación E, mientras que el resto de los modelos predicen con una probabilidad menor que es A, B, C, D, F o G, se le asignará a esa vivienda el certificado E.

Para ello, se genera un notebook en JupyterLab que permite describir, analizar, normalizar y modelar los datos utilizando tanto lenguaje de Python como de R.

A continuación, se describen las partes más relevantes de este notebook:

1. **Importación de librerías.** Se importan diversas bibliotecas y módulos necesarios para realizar análisis de datos, manipulación de datos, visualización y modelado de aprendizaje automático (Machine

Learning). A continuación, se presenta la descripción de las librerías utilizadas clasificadas por su funcionalidad:

a. **Librerías Básicas:**

- *Numpy*. Se utiliza para operaciones matriciales y numéricas eficientes.
- *Matplotlib.pyplot*. Ofrece funciones para la creación de gráficos y visualización de datos.
- *Pandas.* Proporciona estructuras de datos flexibles y herramientas para el análisis de datos.
- *Os* y *sys.* Módulos de sistema operativo y sistema que brindan funciones para interactuar con el sistema de archivos y gestionar rutas.
- *Seaborn*. Se utiliza para mejorar la visualización de los gráficos estadísticos creados con Matplotlib.
- *Warnings.* Se utiliza para gestionar advertencias y controlar su visualización.
- *Itertools.groupby*. Permite agrupar elementos consecutivos en secuencias.
- *IPython.display.* Proporciona herramientas para la visualización en el entorno de Jupyter Notebook.

b. **Operaciones de Texto y Normalización:**

- *Re*. Se utiliza para trabajar con expresiones regulares, que son patrones de búsqueda y manipulación de texto.
- *Unidecode*. Sirve para quitar tildes y reemplazar caracteres especiales en texto.
- *Unicodedata.normalize*. Permite normalizar cadenas de texto mediante la eliminación de diacríticos.

c. **Estadísticas y Análisis:**

- *Scipy.stats*. Proporciona funciones estadísticas, incluyendo pruebas de hipótesis como el test de Fisher y la prueba de chi-cuadrado.
- *Statsmodels.api*. Ofrece clases y funciones para estimación de modelos estadísticos.

d. **Machine Learning (ML):**

- *Sklearn*. Una biblioteca integral de aprendizaje automático que incluye herramientas para clasificación, regresión, clustering y más.
- *LabelEncoder*. Utilizado para codificar etiquetas categóricas en variables numéricas.
- *StandardScaler*. Se utiliza para estandarizar características eliminando la media y escalando a la varianza unitaria.
- *LogisticRegression*. Implementa la regresión logística para problemas de clasificación.
- *KNeighborsClassifier*. Implementa el algoritmo K-Nearest Neighbors para clasificación.
- *DecisionTreeClassifier*. Implementa árboles de decisión para clasificación.

e. **Integración con R:**

- *Rpy2.robjects.* Permite la integración entre Python y R para ejecutar código R desde Python.
- *Pandas2ri*: Facilita la conversión entre objetos de *pandas* y R.

2. **Definición de funciones.** Se realiza de definición de algunas funciones útiles para el análisis y limpieza de datos. Algunas de las definiciones aquí presentes ya han descrito con anterioridad:

a. `describe_columnas`. Toma un dataframe y una lista de nombres de columnas como entrada. Luego, itera sobre cada columna de la lista proporcionada, imprimiendo información relevante sobre sus características. Por cada columna, se muestra el tipo de datos, el número de filas, y el recuento de valores únicos, indicando la frecuencia de cada valor en la columna.

b. `relaciones_vs_target`. Está diseñada para visualizar relaciones entre las características (columnas) de un dataframe y la variable objetivo. Puede recibir un parámetro adicional return_type que por defecto es “axes”, lo cual devuelve los ejes de la figura creada. La función genera subgráficos de dispersión para cada característica en relación con la variable objetivo. Esto facilita la identificación de patrones y relaciones entre las características y la variable objetivo en un conjunto de datos.

c. `represento_doble_hist`. Se encarga de representar dos histogramas en el mismo gráfico. Toma dos conjuntos de datos x_1 y x_0, junto con otros parámetros como el número de bins (n_bins), título, y etiquetas para las clases. Esta función es útil para

comparar la distribución de dos conjuntos de datos, como por ejemplo, la distribución de una característica para dos clases diferentes.

d. `hist_pos_neg_feat`. Se utiliza para representar histogramas comparativos para características (columnas) en función de los valores de la variable objetivo y. Permite visualizar cómo se distribuyen las características para las clases positivas y negativas. La función utiliza la función represento_doble_hist para representar los histogramas de las clases positivas y negativas de cada característica en subgráficos separados.

e. `elimina_diacriticos`. Se encarga de eliminar los diacríticos (acentos, virgulillas, etc.) de una cadena de texto. Utiliza expresiones regulares y la función `normalize` del módulo *`unicodedata`* para lograr la eliminación de diacríticos y normalizar la cadena.

3. **Descripción de los datos.** Se realiza una previa descripción de los datos, con el fin de familiarizarse con los datos a tratar. Se carga un conjunto de datos desde el archivo Parquet "Dataset_model.parquet", se lee y se almacena en un dataframe llamado *XY*. Posteriormente, se imprimen algunas estadísticas descriptivas sobre el conjunto de datos para proporcionar una visión general de su contenido.

   El dataframe *XY* se compone de 42660 filas y 44 columnas. No contiene variables no numéricas y las variables que lo componen son: "ref_cat", "num_m2_const", "cod_prov_INE", "cod_mun_INE", "num_año", "longitud", "latitud", "altitud", "renta_neta_por_hogar", "No tenía ref_cat válida", "Clase-Rural", "Clase-Urbano", "Reforma media", "Reforma mínima", "Reforma total", "Rehabilitación integral", "Sin reformar", "Badajoz", "Cáceres", "ZC-C3", "ZC-C4", "ZC-D3", "ZC-E1", "Agrario", "Almacén - Estacionamiento", "Comercial", "Cultural", "Deportivo", "Edificio singular", "Espectáculos", "Industrial", "Obras de urbanización y jardinería, suelos sin edificar", "Ocio y Hostelería", "Oficinas", "Religioso", "Residencial", "Sanidad y Beneficencia", "A", "B", "C", "D", "E", "F", "G".

   A continuación, se imprime la información sobre los tipos de datos de cada variable presente en el dataframe XY mediante la función `dtypes`. Esto proporciona detalles sobre el formato de datos utilizado para cada variable.

   Se realiza un análisis de los valores nulos en el conjunto de datos utilizando la función `isnull()` y `sum()`, mostrando así la ausencia de valores nulos en cualquiera de los registros que componen al dataframe *XY*.

Finalmente, se presenta un resumen estadístico del conjunto de datos mediante la función `describe()`. Este resumen incluye medidas como la media, la desviación estándar, el mínimo, el 25%, la mediana (50%), el 75% y el máximo para las variables numéricas presentes en el dataframe.

Estas operaciones proporcionan una visión general exhaustiva de la estructura y las características del conjunto de datos, lo cual es fundamental para comprender y analizar eficazmente la información contenida en él.

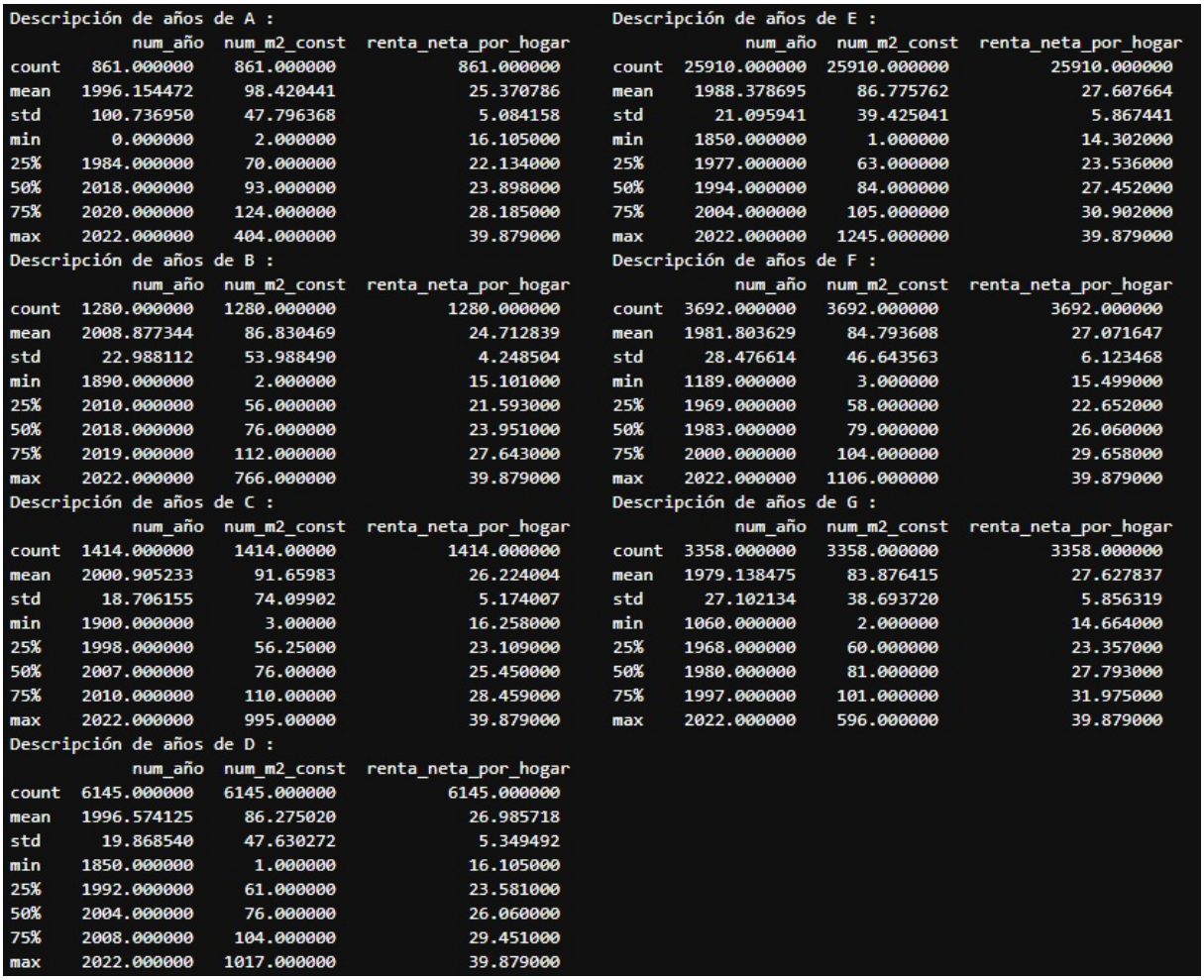
4. **Limpieza y análisis de variables.** El análisis estadístico anterior permitió identificar variables que se presentaban como constantes en el conjunto de datos. Con el objetivo de depurar el dataset, se aplicó la función `drop()` para eliminar dichas constantes, que fueron identificadas como “cod_prov_INE”, “Clase-Rural”, “Clase-Urbano”, “Agrario”, “Espectáculos”, “Religioso” y “Sanidad y Beneficencia”.

   Posteriormente, se lleva a cabo un análisis detallado de las distribuciones de las columnas “num_año”, “num_m2_const” y “renta_neta_por_hogar” en función de cada certificado energético. Este análisis incluye la representación visual de las distribuciones mediante la utilización de histogramas. Para este propósito, se emplean las funciones `subplots` de la librería *`matplotlib.pyplot`* y `histplot()` de la librería *`seaborn`*. Este enfoque visual proporciona una comprensión más profunda de la variabilidad y comportamiento de estas variables en relación con los distintos tipos de certificados energéticos. La eliminación de constantes y el análisis detallado de las distribuciones son pasos cruciales en la preparación de datos para análisis posteriores y modelado predictivo.

   Luego de realizar un análisis exhaustivo de estas columnas, se derivan varias conclusiones fundamentales. En primer lugar, se destaca la presencia de registros atípicos y no representativos en la calificación energética "A", donde la falta de suficientes observaciones contribuye a una variabilidad significativa en las estadísticas descriptivas, como la media y la desviación estándar.

   Además, se identifica una tendencia consistente en los metros cuadrados (m2) para las calificaciones energéticas inferiores a la letra "D", mientras que las calificaciones "A", "B" y "C" muestran una relación inversa entre la calificación y la superficie del edificio. Esta variación en las tendencias complica la predicción de las calificaciones energéticas.

   En este sentido, apuesta por la agrupación de los certificados energéticos en función de su calificación, puesto que puede ser una estrategia eficaz para mejorar la robustez del modelo y solventar la falta de registros de algunas de determinadas calificaciones energéticas. Por ello, se decide agrupar las calificaciones energéticas en cuatro categorías: "A o B", "C o D", "E" y "F o G". Esta clasificación se basa en similitudes observadas en la distribución de los años de construcción, los metros cuadrados y la renta, y busca simplificar la complejidad asociada con las calificaciones individuales.



Descripción de años de A :

	num_año	num_m2_const	renta_neta_por_hogar
count	861.000000	861.000000	861.000000
mean	1996.154472	98.420441	25.370786
std	100.736950	47.796368	5.084158
min	0.000000	2.000000	16.105000
25%	1984.000000	70.000000	22.134000
50%	2018.000000	93.000000	23.898000
75%	2020.000000	124.000000	28.185000
max	2022.000000	404.000000	39.879000

Descripción de años de B :

	num_año	num_m2_const	renta_neta_por_hogar
count	1280.000000	1280.000000	1280.000000
mean	2008.877344	86.830469	24.712839
std	22.988112	53.988490	4.248504
min	1890.000000	2.000000	15.101000
25%	2010.000000	56.000000	21.593000
50%	2018.000000	76.000000	23.951000
75%	2019.000000	112.000000	27.643000
max	2022.000000	766.000000	39.879000

Descripción de años de C :

	num_año	num_m2_const	renta_neta_por_hogar
count	1414.000000	1414.00000	1414.000000
mean	2000.905233	91.65983	26.224004
std	18.706155	74.09902	5.174007
min	1900.000000	3.00000	16.258000
25%	1998.000000	56.25000	23.109000
50%	2007.000000	76.00000	25.450000
75%	2010.000000	110.00000	28.459000
max	2022.000000	995.00000	39.879000

Descripción de años de D :

	num_año	num_m2_const	renta_neta_por_hogar
count	6145.000000	6145.000000	6145.000000
mean	1996.574125	86.275020	26.985718
std	19.868540	47.630272	5.349492
min	1850.000000	1.000000	16.105000
25%	1992.000000	61.000000	23.581000
50%	2004.000000	76.000000	26.060000
75%	2008.000000	104.000000	29.451000
max	2022.000000	1017.000000	39.879000

Descripción de años de E :

	num_año	num_m2_const	renta_neta_por_hogar
count	25910.000000	25910.000000	25910.000000
mean	1988.378695	86.775762	27.607664
std	21.095941	39.425041	5.867441
min	1850.000000	1.000000	14.302000
25%	1977.000000	63.000000	23.536000
50%	1994.000000	84.000000	27.452000
75%	2004.000000	105.000000	30.902000
max	2022.000000	1245.000000	39.879000

Descripción de años de F :

	num_año	num_m2_const	renta_neta_por_hogar
count	3692.000000	3692.000000	3692.000000
mean	1981.803629	84.793608	27.071647
std	28.476614	46.643563	6.123468
min	1189.000000	3.000000	15.499000
25%	1969.000000	58.000000	22.652000
50%	1983.000000	79.000000	26.060000
75%	2000.000000	104.000000	29.658000
max	2022.000000	1106.000000	39.879000

Descripción de años de G :

	num_año	num_m2_const	renta_neta_por_hogar
count	3358.000000	3358.000000	3358.000000
mean	1979.138475	83.876415	27.627837
std	27.102134	38.693720	5.856319
min	1060.000000	2.000000	14.664000
25%	1968.000000	60.000000	23.357000
50%	1980.000000	81.000000	27.793000
75%	1997.000000	101.000000	31.975000
max	2022.000000	596.000000	39.879000

Figura 16. Análisis estadístico de las columnas “num_año”, “num_m2_const” y “renta_neta_por_hogar” para cada certificado energético

5. **Agrupación de variables.** Debido a la limitada cantidad de registros en ciertas certificaciones energéticas, se evidencia una falta de representatividad en los datos. Dada esta situación, se ha optado por realizar agrupaciones estratégicas en variables numéricas clave. Específicamente, se han agrupado la superficie constructiva, el año de construcción y las certificaciones energéticas, clasificándolas en cuatro categorías principales: “A-B”, “C-D”, “E”, y “F-G

   Un aspecto crucial de este análisis se centra en la agrupación del año de construcción. Se han explorado diversos enfoques para determinar la agrupación más efectiva que contribuirá a la distribución equitativa de las categorías de certificación energética, mejorando así la representación de la variable en relación con los certificados energéticos.

   Además, se ha implementado un código específico para calcular percentiles clave (25, 50 y 75) en diferentes conjuntos de datos, considerando las categorías de certificación energética. Este análisis

detallado se realiza sobre la columna “num_año”, proporcionando una visión temporal detallada en cada categoría. La ordenación y selección de valores únicos de los percentiles facilitan la obtención de un conjunto de años ordenado y sin duplicados, mejorando la claridad del análisis.

Un paso adicional implica la creación de grupos de datos basados en intervalos de años, dividiendo el rango total en segmentos que contienen aproximadamente 75,000 registros cada uno. Este enfoque tiene como objetivo ofrecer una representación más manejable y detallada de la distribución temporal en el conjunto de datos, contribuyendo a un análisis más preciso y comprensible.

En otro ámbito, se ha abordado el análisis de los valores de metros cuadrados (m2) en el conjunto de datos. Este proceso incluyó la ordenación de los valores de m2 y la utilización de la función `groupby` para contar los registros en cada categoría de m2. Se estableció un criterio de agrupación que busca crear grupos de alrededor de 8,000 registros por grupo, proporcionando así una visión detallada y clara de la distribución de metros cuadrados en el conjunto de datos. Estos resultados se han almacenado para facilitar la toma de decisiones relacionadas con la agrupación de la variable “num_m2_const”.

Con la ayuda de estos estudios se consigue tras varias iteraciones obtener una distribución que permite aumentar la representatividad de ambas variables. Con el uso nuevamente de las funciones de dibujo de las librerías *`seaborn`* y *`matplotlib`* se presentan a continuación los gráficos.

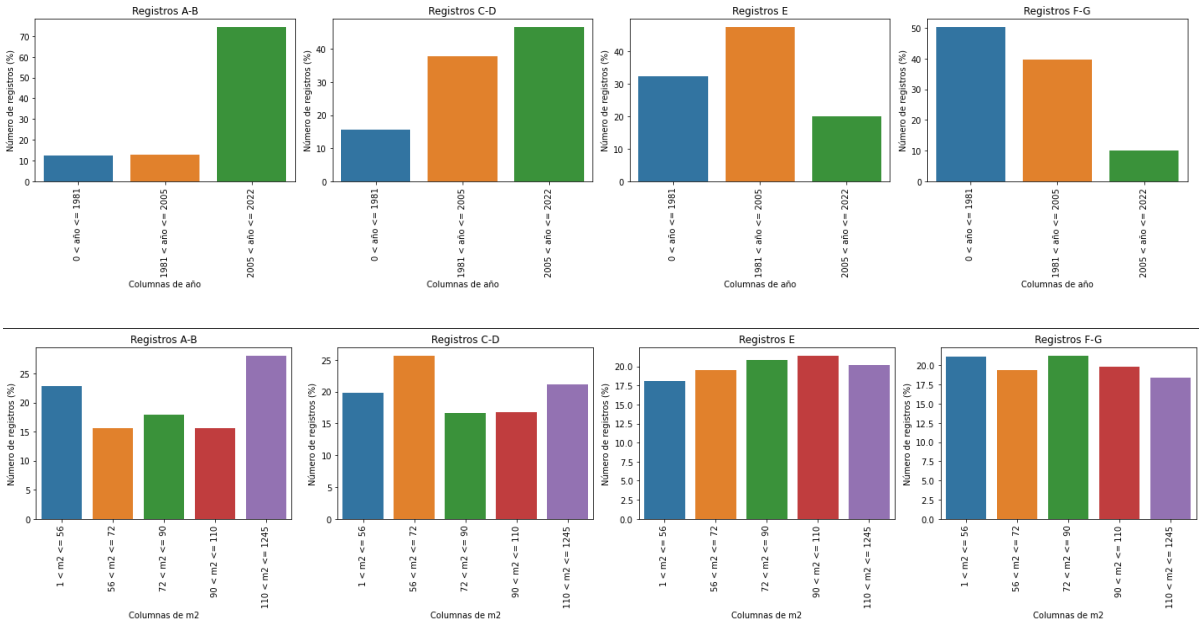


Figura 17. Distribución de las agrupaciones del año y superficie de construcción respecto a la calificación energética

Como resultado de la agrupación estratégica, se logra una distribución porcentual más favorable en las categorías de calificación energética. Se observa que las calificaciones "A-B" predominan en años recientes y en superficies de construcción que se desvían significativamente de la media. En cuanto a las calificaciones "C-D", se empiezan a encontrar en construcciones que datan de 1981 hasta 2005, destacando la predominancia de edificaciones con superficies entre 56 y 72 m^2. La categoría "E" presenta una marcada reducción en registros a partir de 2005, pero en términos de metros cuadrados, se mantiene prácticamente en el mismo rango para todas las agrupaciones. Por último, en la categoría "F-G", se observa un aumento considerable en registros de viviendas anteriores a 1981 y una disminución en registros posteriores a 2005. Además, debido a la antigüedad de estas viviendas, la mayoría de los registros se concentran en superficies constructivas inferiores a 90 m2.

6. **Visualización de los datos.** Se grafican algunas variables de interés en función de la nueva agrupación de las certificaciones energéticas. *Y* se analizan la distribución del resto de variables que componen el conjunto de datos. Para ello se recurren a técnicas de estandarización y normalización de datos, lo que facilita la comparación entre ellas y mejora la convergencia de los algoritmos de aprendizaje automático. Además, esta visualización de datos permite verificar la ausencia de valores atípicos y comportamientos inadecuados de las variables.

   Los gráficos representados en el notebook son los siguientes:

   a. Representación de la distribución de las certificaciones energéticas por coordenadas. utiliza la biblioteca `seaborn` para crear un conjunto de subgráficos que representan la distribución de las certificaciones energéticas en función de las coordenadas (latitud y longitud). El dataframe *XY* se utiliza como fuente de datos, y se grafican las columnas de certificación energética ("A-B", "C-D", "E", "F-G") en cuatro subgráficos separados. Cada subgráfico muestra la distribución espacial de una categoría de certificación energética.

   b. Gráficas de barras para registros de certificaciones y tipos de reforma. Se seleccionan columnas específicas del dataframe *XY* relacionadas con los tipos de reforma ("Reforma media", "Reforma mínima", "Reforma total" y "Rehabilitación integral") y las certificaciones energéticas. Se crean subgráficos de barras para representar el porcentaje de registros en cada tipo de reforma para cada categoría de certificación energética ("A-B", "C-D", "E", "F-G"). La altura de las barras indica el porcentaje de registros con valor 1 en las columnas de tipo reforma para cada categoría de certificación.

   c. Gráficas con valores normalizados. En esta sección, se eliminan algunas columnas del dataframe *XY* y se normalizan las variables restantes ("latitud", "longitud", "altitud" y "renta_neta_por_hogar") para mejorar la comparación y el rendimiento del modelo. Se

generan gráficos de caja para cada variable normalizada, proporcionando una visión clara de la distribución y variabilidad de las variables independientes.

d. Distribuciones de variables independientes. En este código, se crean gráficos de distribución para cada variable independiente normalizada. La biblioteca `seaborn` se utiliza para visualizar la forma y la dispersión de las distribuciones. Cada gráfico representa la distribución de una variable independiente en el conjunto de datos normalizado.

7. **Análisis de correlaciones.** Dado que, en su mayoría, el conjunto de datos está compuesto por variables categóricas se realiza un análisis de correlaciones mediante la prueba exacta de Fisher y el método de Chi-Cuadrado. Esto permite evaluar la asociación o independencia entre de cada variable categórica para cada agrupación de calificación energética. Ambas pruebas permiten analizar la relación entre las variables y determinar si la asociación observada es estadísticamente significativa. Se comparan ambas metodologías y se escoge la que mayores variables dependientes presente.

   Para llevar a cabo la tarea de analizar la correlación entre variables independientes y columnas objetivo mediante la aplicación de la prueba exacta de Fisher, se establece previamente un nivel de significancia, representado por el parámetro `alpha` con un valor de 0,05. A continuación, se utiliza un bucle anidado para iterar sobre cada variable independiente y cada columna objetivo especificada.

   En cada iteración, se crea una tabla de contingencia que resume la frecuencia de ocurrencias conjuntas de las variables en cuestión. Posteriormente, se aplica la prueba exacta de Fisher para evaluar la independencia entre estas variables, generando así el "odds ratio" y el "p-valor" asociados. Los resultados se registran en una lista denominada *resultados*, organizada como diccionarios que incluyen información sobre la variable independiente, la columna objetivo, el "odds ratio" y el "p-valor".

   Luego, se procede a construir un dataframe llamado *fisher_tabla_resultados* a partir de la lista de resultados. Se agrega una columna adicional que indica si el "p-valor" es menor que el nivel de significancia (`alpha`). Posteriormente, se realiza una agrupación de los resultados por columna objetivo, generando un objeto de grupo llamado *tabla_agrupada*.

   En la siguiente fase, se itera sobre los grupos resultantes e imprime tablas específicas para cada columna objetivo. Estas tablas presentan las variables independientes significativas, es decir, aquellas cuyos "p-valor" son inferiores al nivel de significancia establecido, junto con sus respectivos "odds ratio" y "p-valor".

   Por último, se construye un diccionario llamado *fisher_var_ind* que almacena la información sobre las variables independientes significativas para cada columna objetivo. Este diccionario se actualiza

durante el proceso de iteración, proporcionando una estructura organizada que resume las relaciones estadísticamente significativas entre las variables independientes y las columnas objetivo.

De manera análoga, se implementa el código para la prueba de chi-cuadrado. Se establece también un nivel de significancia (`alpha`) de 0.05, y el bucle principal itera sobre cada variable independiente y columna objetivo, creando tablas de contingencia y aplicando la prueba de chi-cuadrado. Los resultados, incluyendo el estadístico chi-cuadrado y el "p-valor", se registran en una lista llamada *resultados*. Se construye un dataframe "chi2_tabla_resultados" y se agrega una columna indicando si el "p-valor" es menor que `alpha`. Se agrupan los resultados por columna objetivo, generando una tabla agrupada. Durante la iteración sobre los grupos, se imprime información detallada sobre las variables independientes significativas para cada columna objetivo, almacenando estos resultados en un diccionario llamado "*chi2_var_ind*".

Finalmente, Se imprimen de los diccionarios *fisher_var_ind* y *chi2_var_ind*, las variables dependientes significativas para cada columna objetivo, según los resultados de las pruebas exacta de Fisher y de chi-cuadrado, respectivamente. Luego, se compara si los diccionarios son iguales.

El resultado obtenido muestra cualquiera de las metodologías es factible para la elección de las variables dependientes, pues se obtienen las mismas variables. Se toma la decisión de escoger la primera de ellas (test de Fisher).

8. **Modelado de los datos.** Se proyectan tres modelos diferentes:

    a. **Regresión logística multivariante.** Utilizada en cuatro modelos idénticos, cada uno enfocado en predecir individualmente una de las variables consideradas. La selección de las variables predictoras se lleva a cabo empleando diferentes estrategias de regresión: pasos sucesivos (stepwise), introducción progresiva (forward), eliminación progresiva (backward), eliminación bidireccional (both). Estos métodos se ejecutan utilizando el entorno R, dado que este ofrece bibliotecas más adecuadas a esta tarea. Partiendo de los resultados de la prueba de exacta de Fisher, se eligen las variables independientes y se evalúan los resultados mediante herramientas como la matriz de confusión, la curva ROC y otros indicadores relevantes como precisión, sensibilidad, especificidad, tasa de error y valor predictivo negativo.

    Además, considerando estas variables se analiza esta misma regresión logística en el entorno de Python con el fin de observar similitudes en el modelo con el modelo de R.

    Para implementar código en lenguaje R directamente en un entorno Jupyter Notebook, se utiliza el comando mágico `%%R` de la interfaz "rpy2". En primer lugar, se realiza una copia del dataframe original *XY* denominada *XY_r*. Luego, se procede a modificar los nombres de las columnas en *XY_r*.

Se convierten los encabezados a minúsculas y se eliminan acentos y caracteres especiales mediante la función `unidecode.unidecode()`. Además, se realizan reemplazos específicos, como cambiar "<" por "..." y "<=" por "...".

Simultáneamente, se trabaja en la adaptación del diccionario *fisher_var_ind*. Se convierten a minúsculas tanto las claves como los valores del diccionario, y se aplican las mismas transformaciones que en *XY_r*. Se utilizan reemplazos específicos para asegurar que los nombres sean interpretables en el entorno R.

A continuación, se incorporan al diccionario *fisher_var_ind_r* las variables cuantitativas "altitud", "latitud", "longitud" y "renta_neta_por_hogar". Esto se logra mediante un bucle que itera sobre las claves del diccionario y añade estas variables a la lista asociada a cada clave.

Finalmente, para facilitar el paso de datos de Python a R, se realiza la conversión del diccionario *fisher_var_ind_r* a un objeto *ListVector*, que es un tipo de dato de R. Esta conversión permitirá utilizar este diccionario en los bloques de código R posteriores para el análisis y modelado de datos.

A continuación, se describe la implementación de las tres metodologías de análisis de los modelos de regresión logística multivariante:

- Método univariante. Se implementa el análisis univariante mediante un bucle en el entorno R. Este análisis se realiza para cada variable dependiente contenida en el diccionario *fisher_RLM_r*.

  El bucle itera sobre cada clave en *fisher_RLM_r*, que representa una variable dependiente. Para cada variable dependiente, se extraen las variables independientes asociadas. Estas variables se convierten en un vector de caracteres utilizando la función `unlist`.

  Para cada combinación de variable dependiente e independiente, se realiza un ajuste de un modelo logístico utilizando la función `glm` (modelo lineal generalizado). La variable dependiente se establece como respuesta, y la variable independiente como el predictor. Se utiliza la familia binomial con enlace "logit", ya que se está tratando con un problema de clasificación binaria.

  Se recopilan los resultados del modelo, incluyendo los coeficientes, intervalos de confianza y significancia estadística. Estos resultados se almacenan en un marco de datos llamado *resultados*. Para cada variable independiente, se añade una fila a este marco de datos.

  La variable temporal se utiliza para seleccionar la segunda fila de los resultados, que contiene información específica sobre la variable independiente. Se agrega una columna adicional

denominada “sig”, que representa el nivel de significancia del coeficiente. Se asignan etiquetas como “***”, ”**”, “*”, o “·” según el valor del p-valor asociado.

Finalmente, se imprime el marco de datos *resultados*, mostrando los coeficientes, intervalos de confianza y etiquetas de significancia para cada variable independiente en relación con la variable dependiente.

- Método Forward. Este análisis se implementa utilizando el método forward y el criterio de Información de Akaike (AIC) para modelos logísticos. Se realiza para cada variable dependiente contenida en el diccionario *fisher_RLM_r*.

  En este contexto, es importante destacar que esta metodología requiere una limpieza previa de las variables dependientes antes de realizar el análisis. Se observa que algunas variables dependientes pueden generar complicaciones que afectan la presentación adecuada de los resultados.

  El bucle itera sobre cada clave en *fisher_RLM_r*, que representa una variable dependiente. Se extraen las variables independientes asociadas a la variable dependiente y se convierten en un vector de caracteres utilizando `unlist`. Además, se inicializa un vector llamado *exclusions* que contendrá las variables que se excluyen del proceso de selección de características. Las variables para excluir se definen según la variable dependiente en consideración. Estas exclusiones se basan en condiciones específicas relacionadas con el tipo de variable dependiente.

  A continuación, se utiliza la función `setdiff` para quitar las variables excluidas del vector de columnas independientes.

  Se obtiene la variable dependiente y se define la fórmula del modelo logístico vacío utilizando la función `reformulate`. Se establece también el horizonte de selección de características utilizando el mismo enfoque.

  Se ajusta un modelo logístico vacío utilizando la función `glm`. Luego, se realiza la selección de características mediante el método forward utilizando el criterio AIC a través de la función `stepAIC`.

  Finalmente, se imprime un resumen del modelo seleccionado utilizando “`print(summary(modlog))`”. Este resumen incluye información sobre los coeficientes, el AIC y los mismos detalles que se incorporaron en la metodología anterior.

- Método backward. En este análisis se implementa el método de selección de variables hacia atrás en el contexto de modelos logísticos. El propósito principal de este método es identificar

las variables independientes más relevantes al ajustar un modelo, comenzando con un conjunto que incluye todas las variables y eliminando iterativamente aquellas que no contribuyen significativamente según el criterio de Información Akaike (AIC). La salida final del proceso se presenta en la consola para su revisión.

Respecto a la arquitectura general del código es idéntica a la utilizada en el método forward, con la única diferencia del parámetro que define la dirección de la función `stepAIC()` que cambia a "backward", indicando la dirección del proceso de selección de características.

- Método bidireccional. Este enfoque combina los métodos forward y backward para encontrar el mejor modelo posible según el criterio de Información Akaike (AIC). Comienza con un modelo vacío y, de manera iterativa, agrega o elimina variables hasta alcanzar el modelo que minimiza el AIC. Los resultados de este proceso se imprimen en la consola para su evaluación.

  La arquitectura general del código es idéntica a la empleada en el método backward, con la única diferencia del parámetro que define la dirección de la función `stepAIC()` que varía a "both", indicando la dirección del proceso de selección de características.

b. **Árbol de decisión.** Esta metodología se lleva a cabo mediante la implementación de un modelo de clasificación mediante árboles de decisión para cada columna objetivo en el conjunto de datos *Y*. En un bucle iterativo, cada columna objetivo se selecciona, y se preparan los datos correspondientes extrayendo las variables independientes y la columna objetivo específica de acuerdo con el diccionario previamente creado (*fisher_var_ind*). Los datos se dividen en conjuntos de entrenamiento y prueba, con un 25% destinado a prueba para evaluar la capacidad del modelo.

   Luego, se procede al entrenamiento de un árbol de decisión utilizando el clasificador `DecisionTreeClassifier`, con una profundidad máxima predefinida de 4.

c. **K-vecinos (KNN).** Esta metodología se lleva a cabo mediante la implementación de un modelo de clasificación basado en el algoritmo de k-vecinos más cercanos (KNN) para cada columna objetivo presente en el conjunto de datos *Y*. En cada iteración del bucle, se preparan los datos seleccionando las variables independientes (*X_current*) y la variable dependiente (*y_current*) asociada a la columna objetivo en cuestión. Posteriormente, se divide el conjunto de datos en conjuntos de entrenamiento y prueba, destinando el 80% de los datos al entrenamiento y reservando el 20% restante para las pruebas.

Se procede a ajustar un modelo KNN utilizando el conjunto de entrenamiento, configurando el parámetro de vecinos más cercanos en 1 (`n_neighbors=1`). Luego, se realizan predicciones tanto de las clases como de las probabilidades asociadas en el conjunto de prueba.

Al final de todos estos modelos, se emplean herramientas uniformes para la evaluación de estos, proporcionando una base consistente y facilitando la comparación entre ellos. Las métricas utilizadas incluyen la matriz de confusión, la curva ROC y otros indicadores esenciales como precisión, sensibilidad, especificidad, tasa de error y valor predictivo negativo. Estas herramientas brindan una visión integral del rendimiento de cada modelo en términos de su capacidad predictiva y discriminativa. Los resultados detallados de estos análisis se examinarán y discutirán en la sección 5.2, ofreciendo una comprensión más profunda de la eficacia y las limitaciones de cada modelo evaluado.

3.7.2 Modelo de superposición de información geográfica en diferentes capas

Para poder detectar las zonas con un alto potencial de renovación energética con diferentes agrupaciones de registros, se usan las capas geométricas de “Municipios_Extremadura”, “MASA” y ”PARCELA”. Para ello, se requiere del uso de un modelo de superposición. Este modelo asigna los registros, almacenados como puntos, del “Dataset_Extremadura_Certificados_Existentes” a las diferentes capas.

El modelo de superposición se incorpora mediante las herramientas de geoprocesamiento que facilita ArcGIS. En concreto se utiliza la operación “Spatial Join” que permite unir datos de dos capas espaciales en función de su proximidad o relación espacial, como la intersección, el contorno, el dentro/fuera, entre otros.

Dado que el objetivo es localizar todos aquellos puntos contenidos en cada elemento poligonal de una capa se utiliza el principio geométrico de “contención espacial”. Este principio geométrico y topológico se basa en la noción de que una entidad espacial está completamente contenida dentro de otra entidad espacial si todos los puntos de la primera entidad están también dentro de la segunda entidad, es decir, si no existen puntos fuera de la segunda entidad que estén dentro de la primera.

El principio de “contención espacial” se clarifica en la siguiente figura.

Figura 18. Contención espacial [17]

Además, la operación de unión se realiza uno a uno. Se une una sola entidad de los datos que se van a unir a una entidad objetivo. Este tipo de operación se escoge debido a que se pretende asignar a cada registro de vivienda la capa poligonal a la que pertenece.

Figura 19. Contención espacial. Unión uno a uno [17]

Gracias a esta función, se consiguen obtener nuevas capas intermedias que se usarán para la asignación de la cantidad de registros y la moda en cada capa con la ayuda del código de Python presente en la sección 3.6.2.

Figura 20. Capas resultantes de la aplicación del modelo Spatial Join

4. Metodología

En esta sección resume la metodología del proyecto por medio de un diagrama de proceso en el que se describen todas las etapas del proyecto.

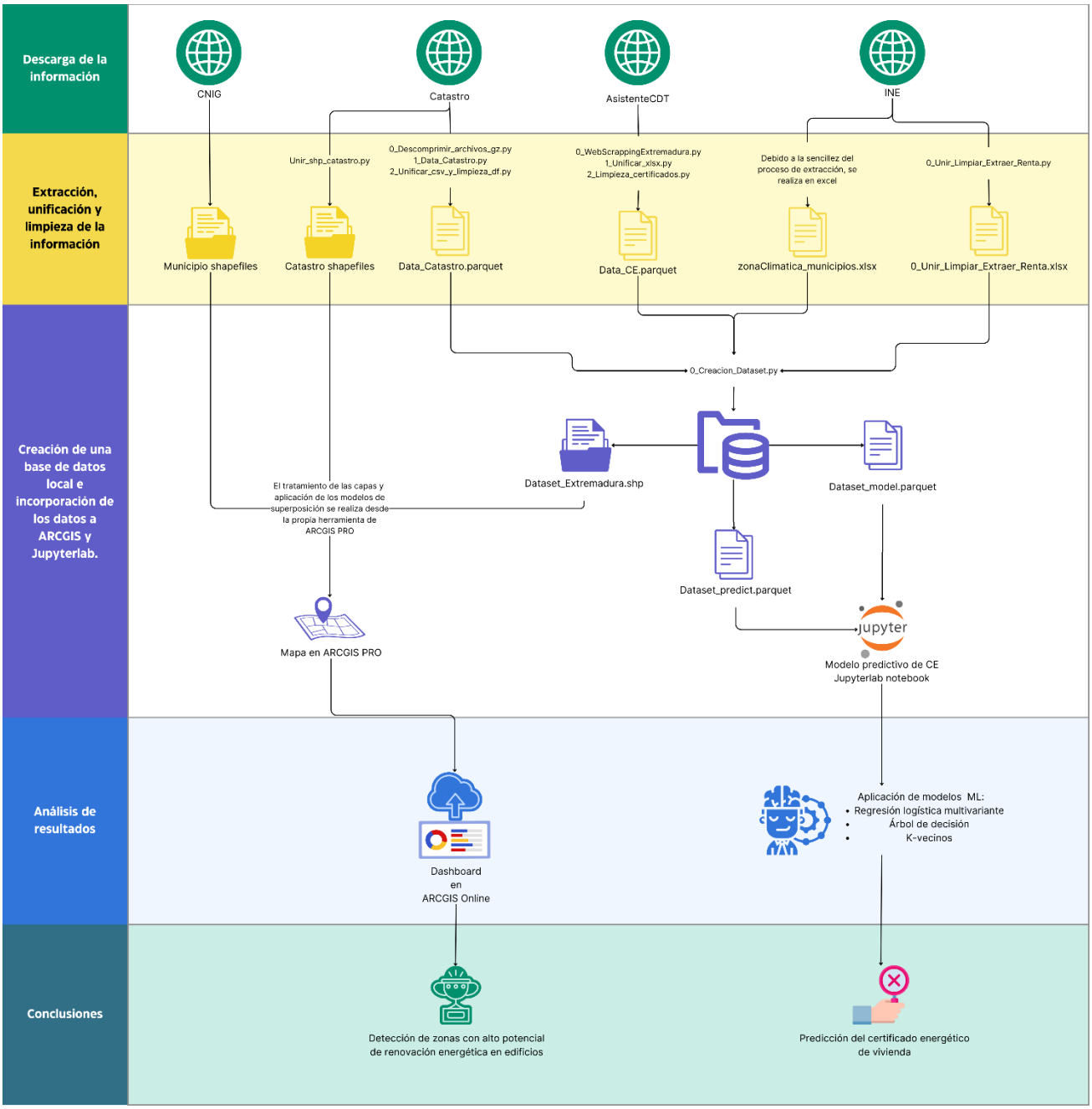


Figura 21. Diagrama de proceso

El proyecto se organiza en cinco etapas distintas: la descarga de información, la extracción, unificación y limpieza de datos, la creación de una base de datos local con la incorporación de datos a ArcGIS y Jupyterlab, el análisis de resultados y las conclusiones. A continuación, se proporciona una breve descripción de estas

etapas, teniendo en cuenta que el detalle de las tres primeras se encuentra en la sección 3, la descripción de la cuarta etapa está en la sección 5 y la última se aborda en la sección 6.

1. **Descarga de la información.** En esta primera fase del proyecto, se procede a la recopilación de datos provenientes de diversas fuentes. Se emplean conjuntos de datos de libre acceso proporcionados por el Centro Nacional de Información Geográfica (CNIG). Estos datos resultan fundamentales para extraer información cartográfica vectorial en formato SHP de los municipios de Extremadura. Además, se accede a la información del Catastro, donde se extrae tanto la información alfanumérica de viviendas y parcelas catastrales de la comunidad autónoma como las capas en formato SHP que incluyen datos sobre masas, parcelas, elementos lineales y elementos constructivos de las parcelas.

   Otra fuente crucial en este proyecto es el AsistenteCDT, el cual facilita información relacionada con la certificación catastral de numerosos registros de vivienda en Extremadura. Por último, se recurre al Instituto Nacional de Estadística (INE) para obtener datos sobre la altitud de los municipios de Extremadura. Estos datos son esenciales para calcular la zona climática a la que pertenece cada municipio, así como la renta neta media por hogar, agrupada a nivel de distrito.

2. **Extracción, unificación y limpieza de la información**. Esta etapa, detallada en la sección 3.1, implica procesos desafiantes de preprocesamiento de datos, demandando un amplio conocimiento de diversas bibliotecas y cierto ingenio para aplicarlas con precisión. Se abordan dos archivos SHP, denominados “Municipio shapefiles” y “Catastro shapefiles”, que contienen toda la información cartográfica espacial para la representación en un mapa de ArcGIS de los datos catastrales.

   En paralelo, se realiza el tratamiento de la información alfanumérica proveniente de Catastro, AsistenteCDT e INE. La extracción, unificación y limpieza de datos varían según el origen de los datos. Por ejemplo, para obtener información alfanumérica de Catastro, se realiza una descompresión previa de archivos en formato GZ, seguida de una extracción de información de archivos en formato CAT a archivos CSV, clasificada en diversas tipologías. Posteriormente, se lleva a cabo la unificación de toda esta información en un archivo PARQUET único denominado “Data_Catastro.parquet”, proceso que implica un análisis profundo de la arquitectura e información de estos archivos CAT mediante diversas fuentes.

   Otro ejemplo de elevada complejidad es la extracción de información de certificados energéticos, almacenada en el archivo “Data_CE.parquet”, utilizando la metodología de Webscraping. Esta técnica requiere un estudio previo de la estructura HTML de la página web, comprendiendo elementos anidados y sus atributos. Además, la problemática principal del rascado de datos es

que incluye posibles errores durante la ejecución del código debido a cambios en la estructura HTML de la web o a los tiempos de carga con los que opera la página.

Adicionalmente, la limpieza de certificados presenta un paso especialmente complejo, ya que su estructura debe asemejarse lo máximo posible a la información alfanumérica catastral para su posterior incorporación en el presente documento.

En cuanto a la información extraída del INE, no requiere una limpieza profunda para la generación subsiguiente de la base de datos local. Las partes más complejas del código en Python se centran en la descompresión de la información y en la preparación y adaptación de la columna clave, que sirve como referencia de unión con la información catastral.

3. **Creación de una base de datos local e incorporación de los datos a** ArcGIS **y Jupyterlab.** Esta fase del proyecto es esencial para consolidar toda la información recopilada. Como se detalla en la sección 3.6, se aprovecha la información procesada en la etapa anterior para generar un nuevo archivo SHP que incluya el conjunto de datos provenientes de Catastro, AsistenteCDT e INE. Además, se crean conjuntos de datos con variables numéricas preparadas para su integración en el entorno de trabajo de Jupyterlab.

   Con la base de datos local debidamente creada, se abordan distintos aspectos vinculados al análisis de datos y la construcción de modelos predictivos en el entorno de Jupyterlab. El proceso se inicia con un análisis exploratorio de datos (EDA), empleando técnicas de visualización para comprender la distribución y correlación de variables. Seguidamente, se aplican pruebas de correlación, como la prueba exacta de Fisher y la prueba de Chi-Cuadrado, para evaluar la asociación entre variables categóricas en diversas agrupaciones.

   Un componente crucial en este proceso es la selección de variables, donde se aplican métodos univariantes, forward, backward y bidireccionales. Estas decisiones se basan en criterios como el Criterio de Información de Akaike (AIC) para identificar las variables más relevantes para los modelos predictivos. Cabe destacar que esta selección se lleva a cabo mediante una traducción de código a R con el apoyo del comando mágico `%%R`, utilizando bibliotecas específicas de este entorno estadístico.

   En cuanto al modelado predictivo, se implementan diversos modelos, entre ellos, regresiones logísticas, árboles de decisión y K-vecinos más cercanos (KNN). La evaluación de estos modelos incorpora métricas esenciales como la curva ROC, el área bajo la curva (AUC), la matriz de confusión y métricas específicas como precisión, sensibilidad, especificidad, tasa de error y valor predictivo negativo.

A lo largo de este proceso, se subraya la importancia crítica de la limpieza y transformación de datos, que incluyen ajustes en los nombres de las columnas y la inclusión de variables cuantitativas en diccionarios. Finalmente, los resultados clave, como las curvas ROC y las representaciones gráficas de árboles de decisión, se visualizan para facilitar la interpretación y comunicación efectiva de los resultados obtenidos.

Por otro lado, los archivos SHP provenientes de "Municipios shapefiles", "Catastro shapefiles" y "Dataset_Extremadura_Certificados_Existentes" son fundamentales para la creación del mapa en ArcGIS Pro Desktop. Es importante destacar que este último archivo, "Dataset_Extremadura_Certificados_Existentes", actúa como el predecesor del "Dataset_Extremadura" debido a ciertos inconvenientes encontrados en la interpretación de datos geoespaciales.

Es relevante mencionar que la utilización de la herramienta ArcGIS Pro Desktop requirió del uso de recursos de capacitación proporcionados por Esri y la lectura de foros donde se compartían experiencias entre usuarios. Sin embargo, el tiempo y esfuerzo invertido para familiarizarse con las características y funcionalidades del software no se extendió excesivamente en el tiempo, dado que se trata de una herramienta visual con una estructura básica similar a Power BI. Con esto me refiero a que implica cargar archivos al proyecto, realizar tareas de limpieza y preparación de datos, y finalmente, presentar los datos de manera gráfica.

El procedimiento para la creación del mapa comprendió la incorporación de capas geométricas ("Municipios_Extremadura", "MASA", "PARCELA" y "CONSTRU"), capas de puntos ("Dataset_Extremadura_Certificados_Existentes") y capas de líneas ("ELEMLIN"). Se requirió solo una breve limpieza de las columnas que componen estos archivos para ofrecer una visión más clara al mostrar la información de cada capa en el mapa resultante. Como se mencionó anteriormente, la incorporación de la columna "Moda_CE" y "Num_reg" se logra mediante la creación de una nueva columna en los archivos resultantes después de implementar el modelo de superposición de información geográfica "Spatial join" a las capas de "PARCELA", "MASA" y "Municipios_Extremadura", cuya incorporación se lleva a cabo con la ayuda de los códigos de Python mostrados en las figuras 14 y 15.

4. **Análisis de resultados.** En esta etapa se parte con el notebook completado, pues incluye los resultados de los modelos de regresiones logísticas, árboles de decisión y K-vecinos más cercanos (KNN), evaluados todo ellos bajo las mismas métricas para facilitar las comparativas entre los modelos para predecir la certificación energética agrupada en las categorías "A-B", "C-D", "E" o "F-G".

Por otro lado, a partir del mapa en ArcGIS Pro Desktop, se crea un Dashboard interactivo centrado en las capas de municipios, masas y parcelas, basándose en la prevalencia de certificados energéticos. Este tablero dinámico se ajusta según la región del mapa, incorporando características como un buscador, leyenda informativa y opciones de interacción. Proporciona una representación visual y dinámica de la información geoespacial, permitiendo a los usuarios explorar la distribución y prevalencia de certificados energéticos.

En términos estadísticos, el tablero muestra datos clave como el número de certificaciones, promedio de año de construcción, superficie promedio de vivienda y renta neta por hogar, brindando una visión completa de estas características en las áreas seleccionadas.

5. **Conclusiones.** Como se mencionó previamente, el proyecto se centra en dos objetivos: la identificación de áreas con potencial de renovación energética y la predicción del certificado energético en viviendas, correspondiendo al objetivo principal y secundario, respectivamente. El primer objetivo se logra con resultados positivos, destacando áreas con certificación energética frecuente a través de un panel interactivo. Sin embargo, el segundo objetivo no se alcanza debido a la limitada cantidad y diversidad de registros disponibles. A pesar de esta limitación, se considera que la herramienta puede llegar a resultar muy favorable a la hora de agilizar decisiones en políticas energéticas y contribuir al desarrollo sostenible en Extremadura. Su utilidad se extiende tanto a nivel gubernamental como empresarial, fomentando inversiones en mejoras energéticas.

5. Resultados

Como se ha mostrado en la sección 4 los resultados finales se dividen en base a dos objetivos: el principal, de detectar zonas de alto potencial de renovación energética; y el secundario, de predecir la certificación energética de vivienda en hogares sin este registro.

5.1 Detección de zonas con elevado potencial de renovación

Se ha desarrollado una herramienta interactiva que facilita la visualización de municipios, masas y parcelas según la prevalencia de registros de certificados energéticos de viviendas presentes en cualquiera de estas tres categorías. La herramienta se presenta en forma de un tablero estadístico dinámico cuyas variables se actualizan automáticamente en correspondencia con la vista mostrada en el mapa. Este mapa dispone de un buscador integrado, una leyenda informativa y amplias opciones de interacción, como desplazamiento y ajuste del nivel de acercamiento.

Las variables estadísticas representadas en el cuadro son:

- Número de certificaciones energéticas desagregadas.
- Promedio del año de las edificaciones.
- Promedio de la superficie de vivienda de las edificaciones.
- Promedio de la renta neta por hogar.
- Número de registros visualizados.

A continuación, se presentan algunas figuras que ilustran el considerable potencial de esta herramienta.

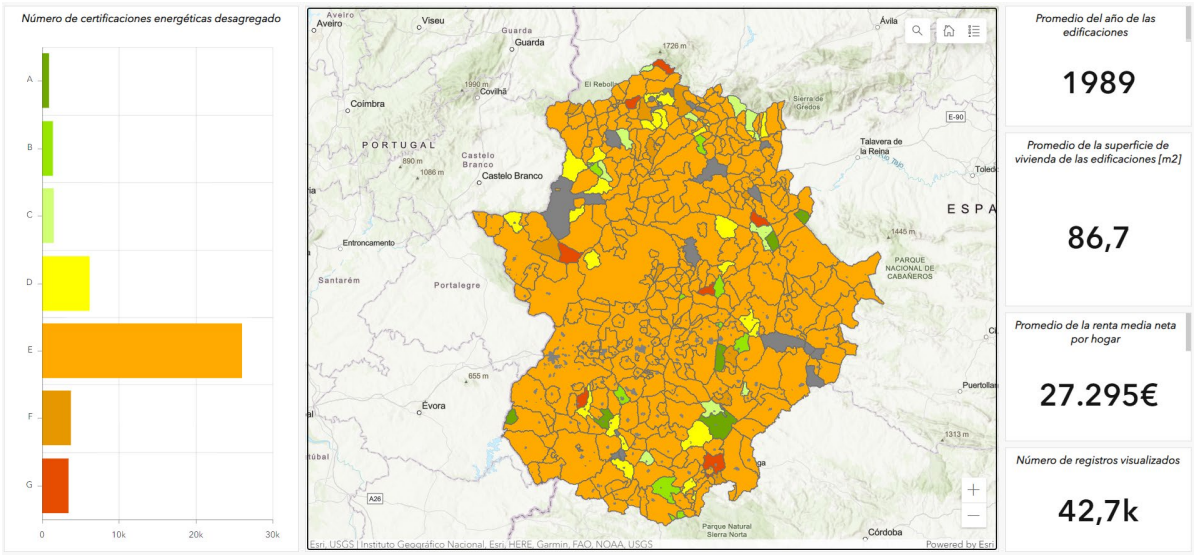


Figura 22. Dashboard. Vista de los municipios de provincia

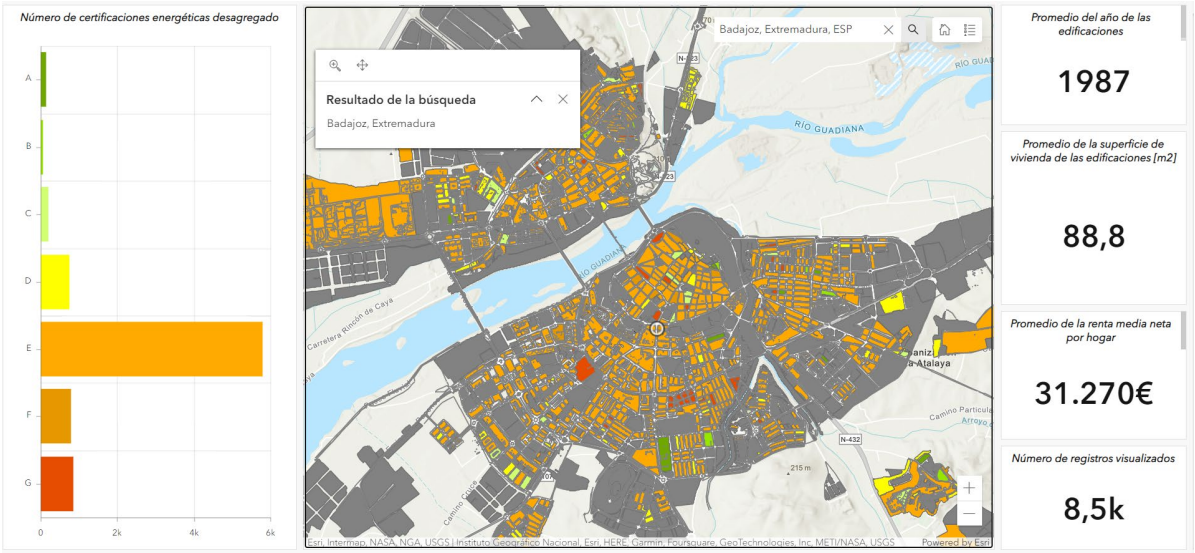


Figura 23. Dashboard. Vista de gran parte de la ciudad Badajoz

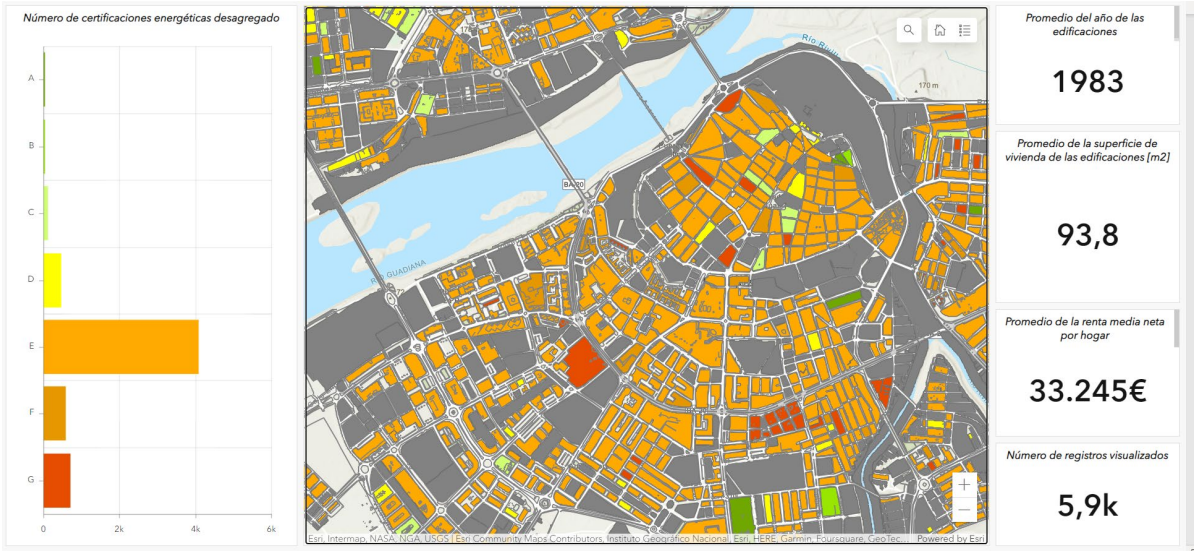


Figura 24. Dashboard. Masas de la ciudad Badajoz

Figura 25. Dashboard. Vista de parcelas con elevado potencial de renovación

Figura 26. Dashboard. Vista de parcelas con bajo potencial de renovación

5.2 Predicción del certificado de eficiencia energética en viviendas

En el notebook de JupyterLab se presentan los resultados derivados de modelos de regresión logística, árboles de decisión y K-vecinos más cercanos (KNN) aplicados en este proyecto. Estos modelos han sido evaluados utilizando métricas uniformes con el propósito de facilitar comparaciones y lograr una predicción efectiva de la certificación energética, categorizada en “A-B”, “C-D”, “E” o “F-G”.

Para evaluar la eficacia de los modelos, se calcula la curva ROC y el área bajo la curva (AUC), proporcionando una medida de la capacidad discriminativa del clasificador. La curva ROC se grafica para visualizar el

rendimiento del modelo en la identificación de verdaderos y falsos positivos. Además, se calcula y muestra la matriz de confusión, detallando los resultados en términos de verdaderos positivos, falsos positivos, verdaderos negativos y falsos negativos.

Además, se calculan diversas métricas de evaluación, tales como precisión, sensibilidad, especificidad, tasa de error y valor predictivo negativo. Estas métricas ofrecen una comprensión completa del rendimiento del modelo para cada columna objetivo. Los resultados son presentados en la consola, permitiendo una evaluación detallada y comparativa de la capacidad predictiva de los modelos en diferentes contextos y para cada variable objetivo en el conjunto de datos.

Tras la exhaustiva evaluación en el entorno de JupyterLab, se concluye que el modelo de regresión logística se presenta como la opción más adecuada para anticipar la certificación energética de viviendas, segmentada en las categorías “A-B”, “C-D”, “E” y “F-G”. Este veredicto se fundamenta en el análisis de cuatro modelos de regresión logística, cada uno utilizando variables significativas seleccionadas mediante la metodología de eliminación bidireccional (“both”) en el contexto de la regresión logística multivariante.

A continuación, se presentan los resultados ejecutados en el entorno de R desde el propio notebook.

```
------------------------------------------------------------------
Variable objetivo:  a.b
------------------------------------------------------------------
[1] 0.07242962
  positivos negativos predichos_positivos predichos_negativos
1      2141     40519                9876                32784
    tp    tn   fp  fn
1 1551 32194 8325 590
Accuracy:  0.791022
Error Rate:  0.208978
Sensitivity:  0.7244278
Specificity:  0.7945408
Precision:  0.1570474
Negative Predictive Value (NPV):  0.9820034
------------------------------------------------------------------
Variable objetivo:  c.d
------------------------------------------------------------------
[1] 0.165062
  positivos negativos predichos_positivos predichos_negativos
1      7559     35101               18817               23843
    tp    tn    fp   fn
1 4990 21274 13827 2569
Accuracy:  0.6156587
Error Rate:  0.3843413
Sensitivity:  0.6601402
Specificity:  0.6060796
Precision:  0.2651857
Negative Predictive Value (NPV):  0.8922535
```

```
------------------------------------------------------------------
Variable objetivo:  e
------------------------------------------------------------------
[1] 0.6038058
  positivos negativos predichos_positivos predichos_negativos
1     25910     16750               27205               15455
    tp   tn   fp   fn
1 18153 7698 9052 7757
Accuracy:  0.6059775
Error Rate:  0.3940225
Sensitivity:  0.7006175
Specificity:  0.4595821
Precision:  0.667267
Negative Predictive Value (NPV):  0.4980912
------------------------------------------------------------------
Variable objetivo:  f.g
------------------------------------------------------------------
[1] 0.1558971
  positivos negativos predichos_positivos predichos_negativos
1      7050     35610               18947               23713
    tp    tn    fp   fn
1 4681 21344 14266 2369
Accuracy:  0.6100563
Error Rate:  0.3899437
Sensitivity:  0.6639716
Specificity:  0.5993822
Precision:  0.2470576
Negative Predictive Value (NPV):  0.900097
```

Figura 27. Resultados modelo de regresión logística en R

Las conclusiones extraídas de la evaluación de los modelos son fundamentales para entender la eficacia de cada enfoque. En primer lugar, es crucial señalar que ninguno de los modelos logra predecir con éxito la variable objetivo, ya que todos presentan una alta tasa de Valor Predictivo Negativo (NPV). Este hallazgo indica

que los modelos tienden a identificar correctamente las instancias que no pertenecen a la clase objetivo, pero tienen dificultades para identificar adecuadamente las instancias que sí pertenecen.

En un análisis más detallado, se observa que, aparentemente, solo se vislumbra la posibilidad de predecir la variable objetivo “E”. Sin embargo, las métricas de precisión y exactitud muestran cifras notablemente bajas, siendo inferiores al 0,65. Esto sugiere que, aunque los modelos pueden identificar algunas instancias de la categoría “E”, su capacidad para hacerlo con precisión y exactitud es limitada.

Para obtener una visión completa y detallada de los resultados de cada modelo analizado, se recomienda revisar el propio notebook, que se encuentra disponible en formatos IPYNB y HTML bajo el nombre de “TFM – Predictor EE”. Estos resultados proporcionan una perspectiva más amplia sobre el rendimiento y las limitaciones de los modelos en el contexto específico de la predicción de certificaciones energéticas.

6. Conclusiones

6.1 Conclusiones técnicas

El proyecto se enfoca en dos objetivos principales: detectar áreas con un alto potencial de renovación energética y predecir el certificado energético de aquellas viviendas con este registro desconocido.

El primer objetivo se aborda de manera versátil y eficiente a través de la creación de un panel de control interactivo. Este panel permite a los usuarios navegar por un mapa digital y visualizar las áreas destacadas por el certificado energético más frecuente, brindando una gran flexibilidad en su uso y funcionalidad.

Por otra parte, el segundo objetivo no se ha podido alcanzar debido a la limitada cantidad y diversidad de registros de viviendas con certificados energéticos disponibles. Además, durante el análisis, no se ha identificado ninguna variable que presente una correlación significativa con la variable objetivo entre las que se han examinado. Por todo ello, no se ha conseguido predecir la certificación de eficiencia energética de las viviendas de las que se carece de esta información.

Respecto a las posibilidades que brinda esta herramienta, son diversas y su impacto potencial es significativo. Si se utiliza de manera efectiva, podría agilizar la toma de decisiones en el ámbito de las políticas energéticas en Extremadura. Esto se traduciría en la revitalización de áreas con baja calificación energética, promoviendo así el desarrollo sostenible y acercando a España un paso más hacia el cumplimiento de los objetivos europeos de consumo y emisiones.

Además, no se limita únicamente al ámbito gubernamental o institucional, sino que también tendría un efecto positivo en el tejido empresarial, tanto en pequeños como grandes negocios relacionados con la construcción y publicidad. Estos actores conseguirían detectar las zonas con un alto potencial de clientes interesados en servicios de renovación de edificios o viviendas.

En resumen, el uso de esta herramienta estimularía la inversión en mejoras energéticas y promovería el desarrollo sostenible, beneficiando a propietarios y a la comunidad en general.

6.2 Conclusiones personales

Esta iniciativa ha representado una invaluable oportunidad para consolidar y poner en práctica los conocimientos adquiridos a lo largo de mi máster. El proceso abarcó una diversidad de aspectos, desde la familiarización y dominio de las herramientas fundamentales como R y Python, hasta la aplicación de análisis exploratorios de datos y la inmersión en el uso avanzado de modelos de aprendizaje automático.

Durante la fase de dominio de herramientas, se profundizó en la comprensión y aplicación de lenguajes de programación cruciales en el ámbito de la ciencia de datos. La habilidad para manipular datos, realizar

operaciones complejas y ejecutar análisis estadísticos se perfeccionó, proporcionando una base sólida para abordar desafíos más avanzados en el proyecto.

La implementación de análisis exploratorios de datos marcó una etapa clave, permitiéndome explorar patrones, identificar tendencias y comprender la estructura de los conjuntos de datos. Esta fase no solo afianzó mi capacidad para trabajar con datos de manera eficiente, sino que también mejoró mi perspicacia para detectar posibles áreas de interés y focalizar los esfuerzos en aspectos clave del proyecto.

En cuanto al uso avanzado de modelos de aprendizaje automático, la aplicación de algoritmos y técnicas avanzadas proporcionó una experiencia valiosa. La capacidad para seleccionar y ajustar modelos según las características específicas de los datos, así como la evaluación crítica de su desempeño, ha sido un componente esencial en el fortalecimiento de mis habilidades técnicas.

Adicionalmente, este proyecto ha representado una notable oportunidad para la adquisición de nuevos conocimientos, destacándose especialmente en el ámbito de la plataforma ArcGIS y la ejecución de tareas en entornos de nube. La creación de un panel de control estadístico integrado en el mapa no solo ha ampliado significativamente mi conjunto de habilidades, sino que también ha proporcionado una experiencia valiosa en la aplicación práctica de tecnologías basadas en la nube.

La familiarización con la plataforma ArcGIS ha sido esencial, permitiéndome explorar y aprovechar sus capacidades para la creación de mapas y el análisis geoespacial. Además, la realización de trabajos en la nube ha proporcionado una visión más profunda de los entornos de trabajo modernos, donde la accesibilidad, la colaboración y la flexibilidad son elementos clave.

La creación del panel de control estadístico no solo implica el dominio de herramientas específicas, sino también la comprensión de cómo integrar y presentar datos de manera efectiva en un formato interactivo. Este aspecto ha ampliado mi perspectiva en términos de visualización de datos y narrativa, contribuyendo a mi capacidad para comunicar información compleja de manera clara y accesible.

En última instancia, esta experiencia ha proporcionado una visión integral del ciclo completo, desde la construcción hasta la evaluación de modelos predictivos. Los beneficios académicos derivados de este proyecto van más allá de la adquisición de conocimientos técnicos, abarcando la capacidad de trabajar en entornos avanzados, comunicar resultados de manera efectiva y adaptar enfoques según las necesidades del cliente.

7. Referencias

[1] BOE, «Real Decreto Legislativo 1/2004, de 5 de marzo, por el que se aprueba el texto refundido de la Ley del Catastro Inmobiliario.» [En línea]. Disponible en: https://www.boe.es/buscar/act.php?id=BOE-A-2004-4163&p=20221224&tn=6

[2] R. con las C. y M. D. núm. 131, de 02 de junio de 2021 Ministerio de la Presidencia, «Real Decreto 390/2021, de 1 de junio, por el que se aprueba el procedimiento básico para la certificación de la eficiencia energética de los edificios».

[3] «Etiqueta del certificado energético». Accedido: 28 de septiembre de 2023. [En línea]. Disponible en: https://tarifasgasluz.com/gestiones/certificado-eficiencia-energetica

[4] G. de E. Ministerio para la Transición Ecológica el Reto Demográfico, «Preguntas frecuentes del Real Decreto 390/2021, de 1 de junio, por el que se aprueba el procedimiento básico para la certificación de la eficiencia energética de los edificios», 2021.

[5] DOE, «DECRETO 115/2018, de 24 de julio, por el que se regulan las actuaciones en materia de certificación de eficiencia energética de edificios en la Comunidad Autónoma de Extremadura y se crea el Registro de Certificaciones de Eficiencia Energética de Edificios.», 2018.

[6] J. de E. Consejería para la Transición Ecológica y Sostenibilidad, «Manual usuario Asistente para la confección de documentación técnica».

[7] «ArcGIS Resource Center». Accedido: 28 de septiembre de 2023. [En línea]. Disponible en: https://resources.arcgis.com/es/help/getting-started/articles/026n00000014000000.htm

[8] «Sede Electrónica del Catastro - Difusión de datos catastrales». Accedido: 28 de septiembre de 2023. [En línea]. Disponible en: https://www.sedecatastro.gob.es/Accesos/SECAccDescargaDatos.aspx

[9] «AsistenteCDT». Accedido: 28 de septiembre de 2023. [En línea]. Disponible en: https://asistenteagile.juntaex.es/AsistenteAGILE/AsistenteInfoCEEE.xhtml;jsessionid=QlJNRwZk8W-6nA-W3eX7sO1r.805699c4-4e4d-3d93-ade6-20e2354c838e)

[10] «Fichero informático de remisión de catastro (bienes inmuebles urbanos, rústicos).»

[11] «Preguntas frecuentes acerca del formato CAT V1.0». [En línea]. Disponible en: http://www.catastro.minhap.es/esp/formatos_intercambio.asp

[12] «Descargas de Cartografía en Formato Shapefile. Manual del Usuario. V1.2».

[13] «Manual de datos de cartografía vectorial (formato shapefile). V2.0.»

[14] «Centro de Descargas del CNIG (IGN)». Accedido: 29 de septiembre de 2023. [En línea]. Disponible en: https://centrodedescargas.cnig.es/CentroDescargas/catalogo.do?Serie=LILIM

[15] «Instituto Nacional de Estadística. (National Statistics Institute)». Accedido: 29 de septiembre de 2023. [En línea]. Disponible en: https://www.ine.es/dynt3/inebase/index.htm?padre=7132

[16] DBHE, «Documento Básico HE».

[17] «How Spatial Join Works in GIS - GIS Geography». Accedido: 1 de octubre de 2023. [En línea]. Disponible en: https://gisgeography.com/spatial-join/