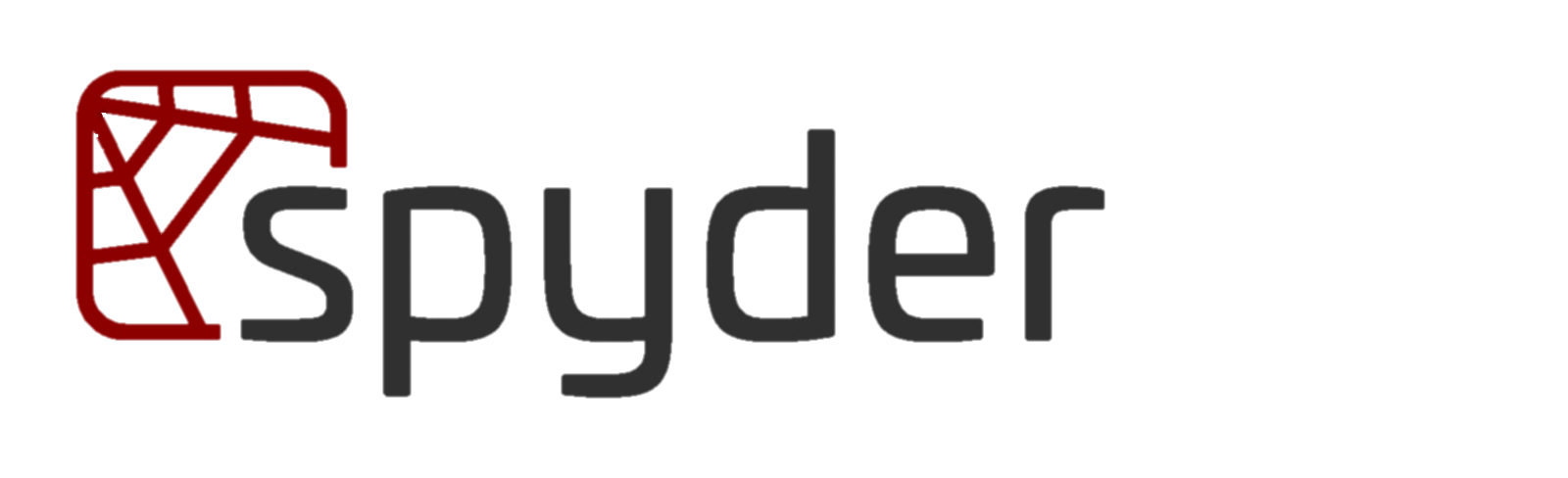
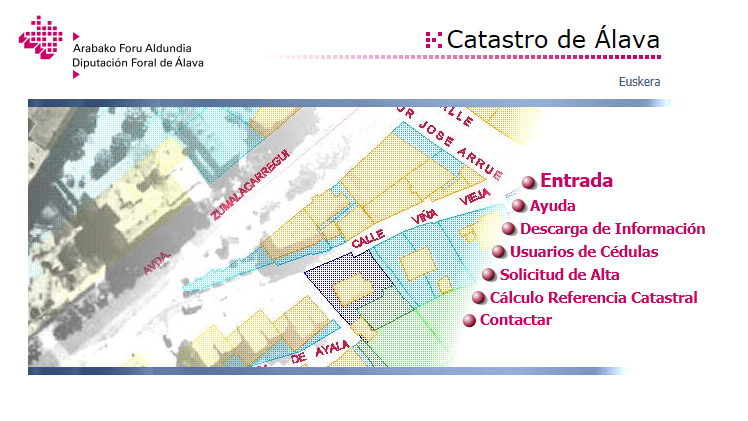
CARACTERIZACIÓN GEOMÉTRICA Y TÉRMICA DEL PARQUE EDIFICADO DE ÁLAVA



**ÍNDICE**

[1. Introducción 3](#_Toc211329963)

[2. Procedimiento 4](#_Toc211329964)

[a. Software necesario 4](#_Toc211329965)

[ QGIS 4](#_Toc211329966)

[ Python-Anaconda-Spyder 4](#_Toc211329967)

[b. Descargar la carpeta comprimida “QGIS\_Araba.zip”. 4](#_Toc211329968)

[c. Descomprimir la carpeta comprimida “QGIS\_Araba.zip”. 4](#_Toc211329969)

[ 0.Códigos 4](#_Toc211329970)

[ 1.Archivos Originales 5](#_Toc211329971)

[ 2.Archivos Modificados 6](#_Toc211329972)

[d. Aplicación al caso de estudio. 6](#_Toc211329973)

[ Archivos catastrales 6](#_Toc211329974)

[ Conseguir archivos del MDSnE2 8](#_Toc211329975)

[ Conseguir archivo climático 8](#_Toc211329976)

[e. Ejecutar el script de QGIS 8](#_Toc211329977)

[f. Ejecutar los modelos pickle de predicción 10](#_Toc211329978)

[3. Resultados 12](#_Toc211329979)

[4. Procesamiento QGIS en detalle 12](#_Toc211329980)

[a. Funciones de los scripts 13](#_Toc211329981)

[ a1.borrar\_cargado 13](#_Toc211329982)

[ a2.borrar\_temporal 13](#_Toc211329983)

[ a3.borrar\_modificado 13](#_Toc211329984)

[ a4.corregir\_combinar 13](#_Toc211329985)

[ a5.CodEdfici 13](#_Toc211329986)

[ a4.corregir\_combinar 13](#_Toc211329987)

[ a4.corregir\_combinar 13](#_Toc211329988)

[ a4.corregir\_combinar 13](#_Toc211329989)

[ a4.corregir\_combinar 13](#_Toc211329990)

[ 14](#_Toc211329991)

[b. Adapatación al parque edificado de Álava 15](#_Toc211329992)

[5. Capacidad de simulación 15](#_Toc211329993)

# Introducción

El presente manual describe el procedimiento para llevar a cabo la caracterización geométrica y térmica del parque edificado residencial de Álava mediante el software QGIS y el posterior uso de modelos de predicción desarrollados en Python.

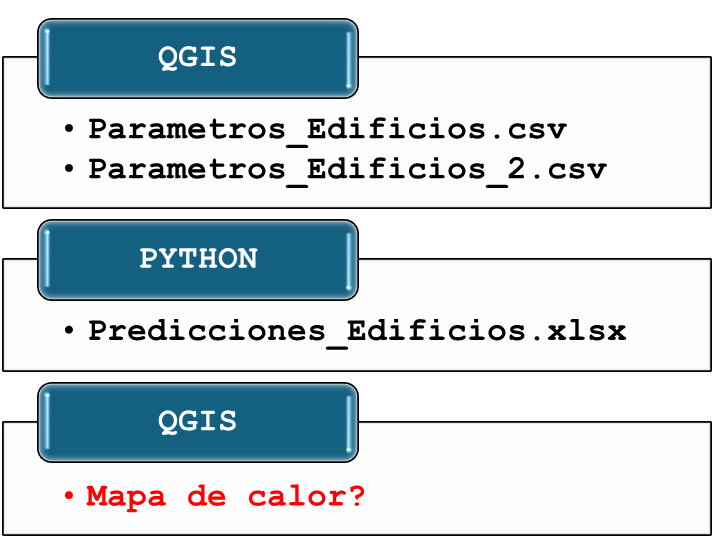
El objetivo principal de este proceso es obtener una estimación de la demanda energética de los edificios residenciales a partir de datos públicos catastrales, modelos digitales de superficie y parámetros climáticos. Para ello, se combina la información geoespacial con técnicas de modelado que permiten calcular indicadores energéticos clave en QGIS, necesarios para la posterior predicción de la demanda.

La metodología propuesta se basa en el uso de datos públicos y en la automatización mediante scripts en Python, lo que simplifica en gran medida el procedimiento y permite que cualquier persona pueda seguir los pasos sin necesidad de conocimientos técnicos avanzados. Aunque el proceso está inspirado en investigaciones previas realizadas en Bizkaia, en este caso se ha adaptado al catastro alavés y a sus características específicas.

Este manual está diseñado como una guía práctica paso a paso, con instrucciones claras y ejemplos visuales que permiten:

* Descargar, organizar y adaptar los archivos necesarios.
* Ejecutar los scripts de procesamiento en QGIS.
* Generar automáticamente tablas de parámetros energéticos de los edificios.
* Aplicar modelos entrenados para obtener predicciones de demanda.

De esta forma, el documento no está dirigido únicamente a investigadores o técnicos, sino que busca ser una herramienta accesible para cualquier persona interesada en analizar y comprender el comportamiento energético de los edificios en Álava, tanto a escala grande como mediana.



# Procedimiento

## Software necesario

Para llevar a cabo este proceso es necesario instalar las dos herramientas principales que usaremos durante el proceso:

### QGIS

QGIS es un software de Sistema de Información Geográfica (SIG) de código abierto, que se utilizará para procesar los datos catastrales y espaciales gracias a su amplia capacidad de manejo de datos vectoriales, ráster y bases de datos.

Es fundamental utilizar la versión QGIS 3.42.3, ya que tanto en versiones anteriores como en algunas posteriores pueden producirse errores de compatibilidad que impedirían ejecutar correctamente ciertos pasos del proceso, impidiendo así conseguir los parámetros necesarios para la predicción energética.

### Python-Anaconda-Spyder

La segunda parte del procesamiento se realiza en Python, donde se ejecutan los modelos entrenados para generar las predicciones de demanda.

Para ello, se recomienda instalar Anaconda, una distribución de Python que simplifica la instalación y gestión de librerías. Con Anaconda es posible crear un entorno controlado con versiones específicas de librerías como scikit-learn o xgboost, lo que resulta fundamental, ya que el uso de versiones diferentes puede provocar errores en la ejecución de los modelos.

Dentro de Anaconda se puede elegir entre distintos entornos de desarrollo integrado (IDE). En este caso, se utiliza Spyder debido a su interfaz sencilla y su facilidad a trabajar con scriptsy.

## Descargar la carpeta comprimida “QGIS\_Araba.zip”.

Una vez tengamos instaladas las herramientas necesarias, el primer paso es descargar la carpeta comprimida que encontraremos \_\_\_\_\_. Esta carpeta comprimida contiene la organización que usaremos para el proceso de caracterización geométrica y térmica de parques edificados de Álava.

## Descomprimir la carpeta comprimida “QGIS\_Araba.zip”.

Descomprimimos la carpeta mencionada. Esta carpeta contiene 3 carpetas en su interior (además de este documento), siguiendo así la siguiente estructura:

### 0.Códigos

En esta primera carpeta, encontramos los scripts que se ejecutarán mediante la consola de Python dentro de la herramienta QGIS. En total encontramos 22 scripts de procesamiento, además de 3 script que su única función es borrar el progreso y modificaciones del proyecto. Los dos últimos scripts tendrían la utilidad de reiniciar el proceso, útil en caso de querer modificarlo. La metodología que componen estos códigos se detalla en el título “*Procesamiento QGIS en detalle*”.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Imagen.n1-Contenido de la carpeta “0.Códigos”

### 1.Archivos Originales

La carpeta de archivos originales será la carpeta en la cual guardaremos los archivos que se requieran analizar en este proceso. En esta carpeta necesitaremos seguir una nomenclatura específica, debido a que los scripts cogen de referencia los archivos debidos a los nombres y formatos. Los archivos requeridos serán los siguientes:

#### Archivos Catastrales

##### Capa “edificio.shp”

##### Capa “division\_altura.shp”

##### Tabla “Edificios\_XXX.csv”

#### Archivos del Modelo Digital de Superficies

##### Rasters “NDSM-EDIFICIOS-XXX.tif”

#### Scripts

##### Código fuente Python para QGIS “Script\_QGIS.py”

##### Código fuente Python para Spyder “Script\_SPYDER.py”

#### Modelos entrenados

##### Modelo de predicción de Temperatura Base “modelo\_TB.pkl”

##### Modelo de cálculo de Demanda “modelo\_Q.ipynb”

#### Archivos Climáticos

##### Tabla “ArchivoClimatico.csv”

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

Imagen.n2-Contenido de la carpeta “1.Archivos Originales”

### 2.Archivos Modificados

La carpeta de archivos modificados se va llenando automáticamente a medida que se ejecutan los diferentes scripts del proceso.

Es importante destacar que, para repetir la simulación o reiniciar el proceso, es necesario borrar previamente el contenido de esta carpeta. La eliminación de los archivos modificados se puede hacer ejecutando el código 25 o de manera manual.

Eliminar estos archivos es imprescindible porque los scripts generan nuevos archivos con nombres específicos. Si estos archivos ya existen, se producirán errores y el proceso no podrá completarse correctamente.

## Aplicación al caso de estudio.

Para aplicar el estudio al caso de interés, sustituiremos ciertos archivos. Los scripts tienen cierta flexibilidad, aun así, se recomienda que se llamen de igual forma a los archivos del ejemplo. Los únicos archivos que requieren ser aplicados al caso de estudio son: archivos catastrales, rasters del Modelo Digital de Superficies Edificación y el archivo climático.

### Archivos catastrales

Como bien se menciona anteriormente, desde el catastro debemos conseguir las capas de edificio y división de altura, además de las tablas de parámetros de los edificios. Para esto, se usará la página web del [Catastro de Araba](https://catastroalava.tracasa.es/descargas/?lang=es).

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Imagen.n3-Pagina web del Catastro Alavés

La propia pagina web, claisifica toda la información en dos diferentes carpetas “XXX\_CatastroAlava” y ”DatosAlfanumericos”. Para el proceso, únicamente son de interes dos capas de la carpeta “CatastroAlava” llamadas edificio.shp y division\_altura.shp, ademas de una tabla nombrada Edificios\_XXX.csv dentro de ”XXX\_DatosAlfanumericos”.

Cabe destacar que el catastro alaves da la opción de escoger el municipio del que se quiere descargar la informacion catastral, pero la realidad es que en la mayoría de los casos el caso de interes es menor a un municipio entero o incluso se puede llegar a encontrar entre dos municipios. Para esos casos, se necesitaría un proceso no automatizado, que se deberá seguir previo a cualquier simulacion de scripts.

En el caso de que los edificios de interés se encuentren entre varios municipios, se podrá usar la herramienta “Combinar capas vectoriales”. Esta función combina distintas capas vectoriales en una sola. Tras utilizar la función, se podrá exportar la nueva capa reusltante y nombrar de igual manera.

En cambio, si el caso de estudio es más pequeño que un solo o varios municipios enteros, se podrá aplicar la función de “Filtro de capa”. Esta función permite aplicar ciertos filtros para obtener las capas con los valores de los edificios de interes (la herramienta de filtro de capa ayuda a filtrar por municipio, polígono, parcela, subparcela y edificio). Además de esa función, se podrán escoger las capas de edificios y divisones que no se quieran analizar y eliminar, de manera totalmente manual con “Seleccionar objetos espaciales a mano alzada” y “Borra lo seleccionado”.

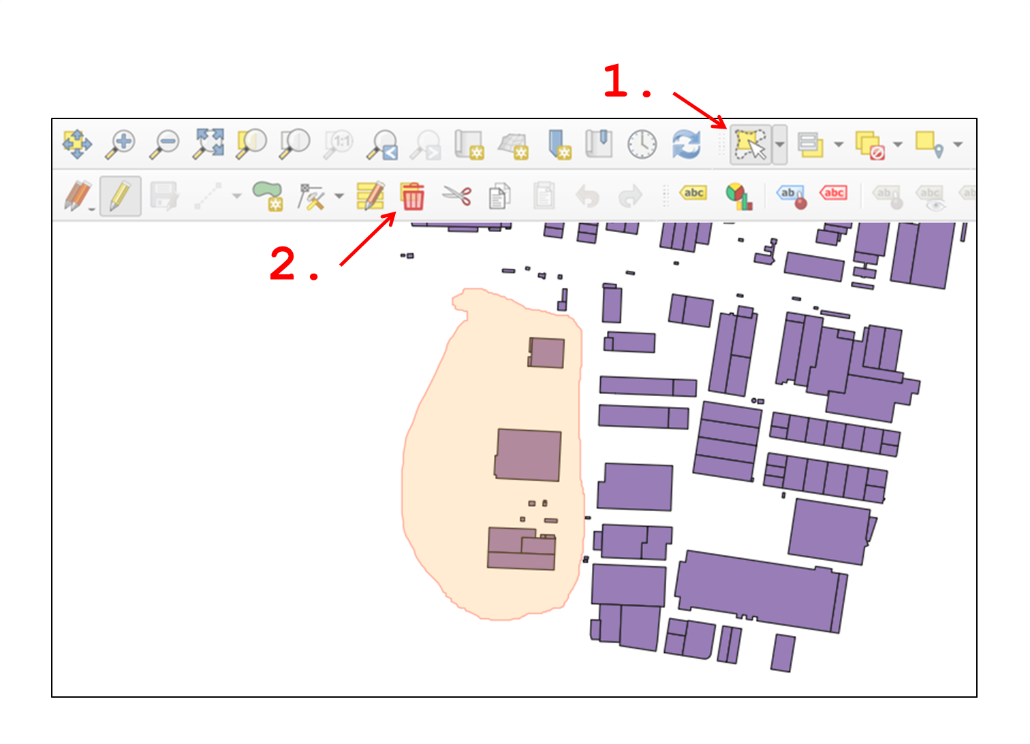


Imagen.n4-Eliminar objetos no deseados de manera manual en QGIS

### Conseguir archivos del MDSnE2

Para conseguir unos datos de altura de edificio lo más realistas posibles, se utiliza la capa Modelo Digital de Superficies Edificación (MDSnE2,5 2ª Cobertura) del [CNIG](https://centrodedescargas.cnig.es/CentroDescargas/buscarGeom).

Dependiendo del caso de estudio, puede ser necesario trabajar con varias capas ráster si los edificios de interés se encuentran en distintos rasters. Sin embargo, no es necesario combinarlas manualmente (como con las capas vectoriales), ya que el proceso está automatizado y coge de referencia la capa vectorial “edificio.shp”.

### Conseguir archivo climático

¿?????

## Ejecutar el script de QGIS

Una vez aplicados los archivos al caso de interes y guardarlos dentro de la carpeta de “Archivos Originales”, se podrán ejecutar “Script\_QGIS.py”.

Para eso, abrimos el software QGIS y pulsamos “CTRL+ALT+P” para abrir la consola de Python. Después, escogemos el script dentro de “Abrir script…” y una vez escogido se podría ejecutar el siguiente código:

import os

import time

import sys

from PyQt5.QtWidgets import QFileDialog, QApplication

# --- Inicializar aplicación de PyQt ---

app = QApplication.instance()

if not app:

app = QApplication(sys.argv)

# --- Seleccionar carpeta de códigos ---

ruta\_codigos = QFileDialog.getExistingDirectory(

None,

"Selecciona la carpeta donde están los códigos",

os.path.expanduser("~")

)

if not ruta\_codigos:

raise ValueError("No se seleccionó ninguna carpeta")

# --- Lista de archivos .txt ---

archivos = [f for f in os.listdir(ruta\_codigos) if f.endswith(".txt")]

# --- Guardar resultados en lista ---

tiempos = []

# --- Ejecutar cada archivo con medición de tiempo ---

for archivo in archivos[0:22]:

# for archivo in archivos[22:25]:

ruta\_completa = os.path.join(ruta\_codigos, archivo)

print(f"⏳ Ejecutando: {archivo}")

with open(ruta\_completa, "r", encoding="utf-8") as f:

codigo = f.read()

try:

inicio = time.time()

exec(codigo)

fin = time.time()

duracion = fin - inicio

print(f"✅ Ejecutado en {duracion:.2f} segundos\n")

tiempos.append(round(duracion, 2)) # redondeado a 2 decimales

except Exception as e:

print(f"❌ Error al ejecutar {archivo}: {e}\n")

tiempos.append(None)

# --- Escribir los timepos ---

# Usamos "-" como separador

texto\_final = "-".join(str(t if t is not None else "ERR") for t in tiempos)

print(texto\_final)

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

Imagen.n5-Ejecutar script en la consola de Python de QGIS

Tras pulsar “Ejecutar script”, el ordenador nos pedirá escoger el directorio de las diferentes carpetas que tenemos. Tras eso, ejecutará todos los pasos del proceso y tras un tiempo nos generará y cargará la tabla de parámetros energéticos de los edificios llamada “Parametros\_Edificios.csv” y “Parametros\_Edificios\_2.csv”. Estos dos archivos serán necesarios para hacer las posteiores predicciones mediante modelos entrenados.

Dentro del script existe una parte (aquí en color rojo) que nos facilita el volver a empezar la ejecución sin problemas debido a que su función es quitar las capas cargadas y borrar los archivos cargados en QGIS o descargados en la carpeta de archivos modificados. Para usar esa función, debemos poner la parte verde como comentario (mediante #) y descomentar la parte roja, resultando en algo así:

# for archivo in archivos[0:22]:

for archivo in archivos[22:25]:

Una vez se consigan las dos tablas de parámetros, se puede cerrar tanto la consola de Python de QGIS, como el propio software QGIS.

## Ejecutar los modelos pickle de predicción

Una vez terminada la labor de QGIS, procederemos a ejecutar los pickle mediante Python. Para eso, necesitaremos tener instalado previamente Anaconda y generar un entorno con unas librerías con versiones específicas mediante la aplicación Anaconda Prompt.

Para generar el entorno que requiere el modelo, en Anaconda Prompt se ejecutaran las siguientes ordenes:

* conda create -n modelo\_antiguo python=3.9 scikit-learn=0.24.2 xgboost=1.7 joblib
* conda activate modelo\_antiguo
* conda install spyder
* conda install pandas
* conda install -c anaconda xlsxwriter
* conda install -c anaconda openpyxl
* spyder

Tras esto, se genera el entorno apto y abre Spyder, que es el entorno de desarrollo integrado para Python escogido. Una vez se abra Spyder, tendremos que abrir el segundo script que encontramos en la carpeta de archivos originales, “Script\_SPYDER.py”, y aparecerá el siguiente código:

import os

import pandas as pd

import numpy as np

import xlsxwriter

import joblib

import xgboost as xgb

from PyQt5.QtWidgets import QApplication, QFileDialog

import sys

app = QApplication.instance()

if app is None:

app = QApplication(sys.argv)

ruta\_original = QFileDialog.getExistingDirectory(

None,

"Selecciona la carpeta de archivos originales"

)

if not ruta\_original:

print("❌ No se seleccionó la carpeta de archivos originales")

exit()

ruta\_modificado = QFileDialog.getExistingDirectory(

None,

"Selecciona la carpeta donde guardar archivos modificados"

)

if not ruta\_modificado:

print("❌ No se seleccionó la carpeta de archivos modificados")

exit()

ruta\_pickle = os.path.join(ruta\_original, "modelo\_TB.pkl")

grid = joblib.load(ruta\_pickle)

xgb\_model = grid.best\_estimator\_

ruta\_json = os.path.join(ruta\_modificado, "modelo\_TB.json")

booster = xgb\_model.get\_booster()

booster.save\_model(ruta\_json)

booster\_mod = xgb.Booster()

booster\_mod.load\_model(ruta\_json)

ruta\_csv = os.path.join(ruta\_modificado, "Parametros\_Edificios.csv")

if not os.path.exists(ruta\_csv):

print(f"❌ No se encontró el CSV de prueba en {ruta\_csv}")

exit()

for sep in [';', ',']:

X\_test = pd.read\_csv(ruta\_csv, sep=sep)

if len(X\_test.columns) > 1:

break

else:

raise ValueError("No se pudo detectar correctamente el separador del CSV.")

if "CodEdifici" in X\_test.columns:

X\_test\_model = X\_test.drop(columns=["CodEdifici"])

else:

X\_test\_model = X\_test.copy()

for col in X\_test\_model.columns:

if X\_test\_model[col].dtype == 'object':

try:

X\_test\_model[col] = pd.to\_numeric(X\_test\_model[col], errors='coerce')

except:

pass

for col in X\_test\_model.select\_dtypes(include='object').columns:

X\_test\_model[col] = X\_test\_model[col].astype('category')

dtest = xgb.DMatrix(X\_test\_model, enable\_categorical=True)

preds = booster\_mod.predict(dtest)

dframe = pd.DataFrame()

if "CodEdifici" in X\_test.columns:

dframe["CodEdifici"] = X\_test["CodEdifici"]

dframe["TB Model Prediction (ºC)"] = preds

ruta\_pred = os.path.join(ruta\_modificado, "Parametros\_Edificios\_TB.csv")

dframe.to\_csv(ruta\_pred, index=False)

print(f"✅ Predicciones guardadas en: {ruta\_pred}")

tipo = "TB"

model = "with HDD"

alg = "XGBoost"

tiempo = "anual"

version = "V2"

archivo\_climatico = os.path.join(ruta\_original, "ArchivoClimatico.xlsx")

df\_temp = pd.read\_excel(archivo\_climatico, engine="openpyxl")

df\_temp = df\_temp.set\_index('Date/Time')

ts\_temp = df\_temp['OutTemp']

hdd\_tot = []

base\_temp = dframe["TB Model Prediction (ºC)"]

for tb in base\_temp:

df\_temp["HDD"] = tb - ts\_temp

df\_temp["Logico"] = np.where(df\_temp["HDD"] < 0, 0, 1)

df\_temp["HDD2"] = df\_temp["HDD"] \* df\_temp["Logico"]

hdd = df\_temp["HDD2"].sum()

hdd\_tot.append(hdd / 24)

archivo\_hdd = os.path.join(ruta\_modificado, f'Parametros\_TB\_HDD.xlsx')

workbook = xlsxwriter.Workbook(archivo\_hdd)

worksheet1 = workbook.add\_worksheet("Resultados")

header = ["TB", "HDD"]

for col, h in enumerate(header):

worksheet1.write(0, col, h)

for row, val in enumerate(base\_temp, start=1):

worksheet1.write(row, 0, float(val))

for row, val in enumerate(hdd\_tot, start=1):

worksheet1.write(row, 1, val)

workbook.close()

print("✅ HDD value export is done")

archivo\_UA = os.path.join(ruta\_modificado, "Parametros\_Edificios\_2.csv")

df\_UA = pd.read\_csv(archivo\_UA, sep=';')

sigma = df\_UA["UA + ρVcp (KW/K)"]

PDemand = np.multiply(sigma, hdd\_tot)

PredDemand = np.multiply(PDemand, 24)

archivo\_final = os.path.join(ruta\_modificado, f'Predicciones\_Edificios.xlsx')

dframe["UA + ρVcp (KW/K)"] = sigma

dframe["HDD (ºC·days)"] = hdd\_tot

dframe["Prediction (kWh)"] = PredDemand

dframe.to\_excel(archivo\_final, index=False)

print(f"✅ Cumulative prediction result is exported in: {archivo\_final}")

Una vez ejecutado este script, el propio Spyder pedirá situar los directorios, de nuevo, de las carpetas de los archivos originales y modificados. A partir de ahí, ejecutará el script generando predicciones de temperatura base y estimaciones de demanda en cada edificio.

Una vez se genere la tabla de las demandas en la carpeta de archivos modificados, ajo el nombre de “Predicciones\_Edificios.xlsx”, se podría dar por acabado el proceso.

# Resultados

Dentro del resultado de “Predicciones\_Edificios.xlsx”, encontramos 5 parámetros de interés:

* ***CodEdifici:*** Es el código elegido como identificador (ID o primary key) y se compone a base de concatenar los números o letras de los siguientes datos del catastro: "Municipio" || "Poligono" || "Parcela" || coalesce("Subparce", 1) || "Edific".
* ***TB Model Prediction (ºC):*** Este parámetro hace referencia a la temperatura base predicha por el modelo entrenado.
* ***UA + ρVcp (KW/K):*** La suma de la transmitancia térmica global del edificio (UA) y capacidad térmica del flujo de ventilación (ρVcp).
* ***HDD (ºC·days):*** HDD hace referencia a Heating Degree Days o Grados-Día, al final, este parámetro es un indicador climático que mide cuánto y durante cuánto tiempo la temperatura exterior está por debajo de la temperatura base.
* ***Prediction (kWh):*** Este último parámetro representa la predicción anual de la demanda calorífica del edificio expresada en kilovatios-hora (kWh). Es un valor que integra tanto las características constructivas del edificio (aislamiento, calidad de la envolvente, ventanas, puentes térmicos, etc.) como las condiciones climáticas de la zona en la que se ubica. Se podría decir que es la razón de este procesamiento.

Tabla

El contenido generado por IA puede ser incorrecto.

Imagen.n6-“Predicciones\_Edificios.xlsx”

# Procesamiento QGIS en detalle

En este manual del proceso, se ignoran todos los subprocesos que se ejecutan en los códigos de QGIS, pero en caso de interes, es necesario dejar claro que se replica la metodologia escogida para el articulo “[*The Open Data Potential for the Geospatial Characterisation of Building Stock on an Urban Scale: Methodology and Implementation in a Case Study*](http://hdl.handle.net/10810/64488)”. En ese articulo, se proponen ciertos pasos y estimaciones basadas en el parque edificado de Bizkaia y su catastro, pero en este caso, se trasladan hacia el parque edificado de Álava. La explicación del procesamiento se detalla mediante las funciones que recoge cada bloc de notas o conjunto de scripts:

## Funciones de los scripts

### a1.borrar\_cargado

#### Quita las capas cargadas en QGIS.

### a2.borrar\_temporal

#### Borra las capas cargadas temporales en QGIS.

### a3.borrar\_modificado

#### Borra los archivos de la carpeta: “2.Archivos Modificados”.

### a4.corregir\_combinar

#### Corrige las geometrías de las capas “.shp” y las llama igual pero con el sufijo “\_corregida”.

#### Combina todos los rasters existentes cn formato “.tif” en el raster “MDS.tif”.

### a5.CodEdfici\_1

#### Crea el campo identificador CodEdifici en la capa “edificio\_corregida.shp”.

#### Crea el campo identificador CodEdifici en la tabla “Edificios\_XXX.csv”.

#### Generá una tabla llamada “Edificios.csv” con los datos de interes de la tabla “Edificios\_XXX.csv”.

### a6.cargar\_modificado

#### Carga todos los archivos modificados.

### a7.centroide

#### Genera centroides en capa “division\_altura\_corregida.shp”, creando así la capa "division\_altura\_punto.gpkg".

### a8.CodEdifici\_2

#### Aplica la función “Unir atributos por localización” para copiar el campo CodEdfici desde la capa “edificio\_corregida.shp” hacia los puntos, creando así la capa “division\_altura\_punto\_CodEdifici.gpkg”.

#### Aplica la función “Unir atributos por localización” para copiar el campo CodEdfici desde la capa de puntos “division\_altura\_punto\_CodEdifici.gpkg” hacia la capa “division\_altura\_corregida.shp” creando la nueva capa “division\_altura\_0.gpkg”.

### a9.campos\_1

#### Se usa la función “Unión” para unir los campos de interes desde la tabla “Edificios.csv” hacia “division\_altura\_0.gpkg” usando el campo CodEdifici como clave común, creando así a la capa “division\_altura\_1.gpkg”.

#### Se crean los campos ANO\_est y ALT\_calc en la capa “division\_altura\_1.gpkg”.

### b10.geomterias

#### Quita las capas cargadas en QGIS.

### a2.borrar\_temporal

#### Borra las capas cargadas temporales en QGIS.

### a3.borrar\_modificado

#### Borra los archivos de la carpeta: “2.Archivos Modificados”.

### a4.corregir\_combinar

#### Corrige las geometrías de las capas “.shp” y las llama igual pero con el sufijo “\_corregida”.

#### Combina todos los rasters existentes cn formato “.tif” en el raster “MDS.tif”.

### a5.CodEdfici\_1

#### Crea el campo identificador CodEdifici en la capa “edificio\_corregida.shp”.

#### Crea el campo identificador CodEdifici en la tabla “Edificios\_XXX.csv”.

#### Generá una tabla llamada “Edificios.csv” con los datos de interes de la tabla “Edificios\_XXX.csv”.

### a6.cargar\_modificado

#### Carga todos los archivos modificados.

### a7.centroide

#### Genera centroides en capa “division\_altura\_corregida.shp”, creando así la capa "division\_altura\_punto.gpkg".

### a8.CodEdifici\_2

#### Aplica la función “Unir atributos por localización” para copiar el campo CodEdfici desde la capa “edificio\_corregida.shp” hacia los puntos, creando así la capa “division\_altura\_punto\_CodEdifici.gpkg”.

#### Aplica la función “Unir atributos por localización” para copiar el campo CodEdfici desde la capa de puntos “division\_altura\_punto\_CodEdifici.gpkg” hacia la capa “division\_altura\_corregida.shp” creando la nueva capa “division\_altura\_0.gpkg”.

### a9.campos\_1

#### Se usa la función “Unión” para unir los campos de interes desde la tabla “Edificios.csv” hacia “division\_altura\_0.gpkg” usando el campo CodEdifici como clave común, creando así a la capa “division\_altura\_1.gpkg”.

#### Se crean los campos ANO\_est y ALT\_calc en la capa “division\_altura\_1.gpkg”.

## Adapatación al parque edificado de Álava

# Capacidad de simulación

El procesamiento del parque edificado y la predicción de la demanda se apoyan en dos subprocesos desarrollados en diferentes softwares. De ellos, el más exigente a nivel computacional es QGIS.

Aunque QGIS permite simular conjuntos de gran escala, en la práctica puede presentar limitaciones en la ejecución de ciertos scripts cuando el tamaño del caso de estudio es demasiado ambicioso. Sin embargo, para el propósito planteado en este proyecto, es decir, pequeña y mediana escala, el flujo de trabajo se completa satisfactoriamente y ofrece resultados consistentes en un corto tiempo.

Como referencia, en la figura siguiente se recoge el tiempo de simulación obtenido en distintos tamaños de casos analizados, ejecutados en un ordenador portátil de gama media (Lenovo IdeaPad 3 con procesador AMD Ryzen 7, serie 5000):

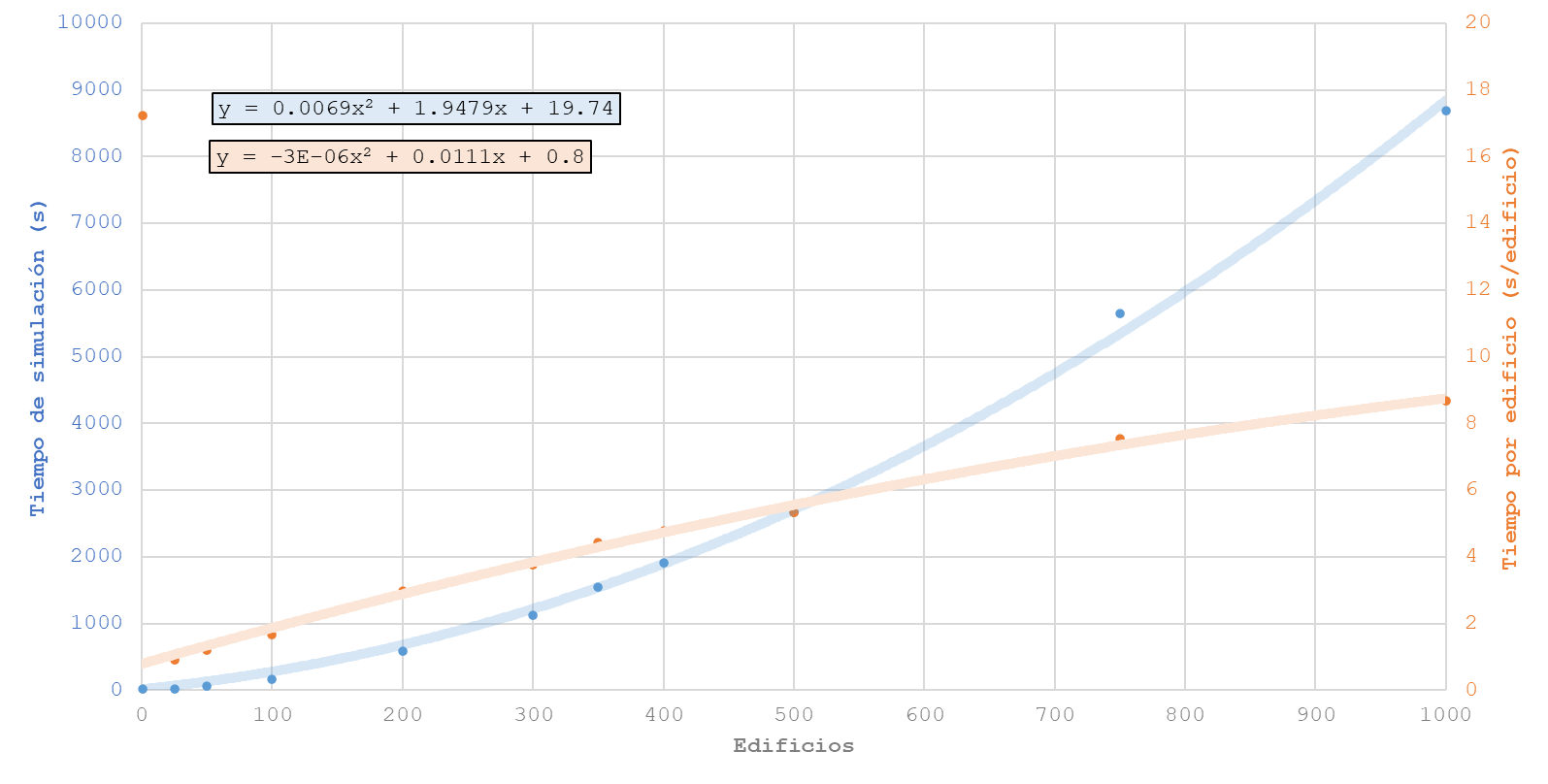


Imagen.n7-Gráfico de tiempo de simulación respecto a cantidad de edificios

Tal como se observa en el gráfico, al aumentar la cantidad de edificios analizados simultáneamente, el tiempo de ejecución también crece, aunque no de manera proporcional. Esto se debe a que la capacidad computacional del ordenador se reparte entre los procesos, por lo que resulta más eficiente dividir el caso de estudio en rangos, reduciendo así el tiempo total de simulación.

Este comportamiento se refleja en el parametro de *tiempo de simulación por edificio*, indicado en color naranjbnm,a. Ese dato deja claro que cuanto menor es el tamaño del caso de estudio, menor es el tiempo de simulación por unidad. Por ejemplo, para 25 edificios el tiempo unitario de cada edificio es inferior a 1 s, mientras que al simular 1000 edificios, cada uno tardaría aproximadamente 8,7 s en completarse.