Tipología y Ciclo de Vida de los Datos

Máster de Ciencia de Datos de la UOC

Práctica 1

Enrique Javier Andrés Orera

Sergio Fernández Bertolín

1. **Contexto. Explicar en qué contexto se ha recolectado la información. Explique por qué el sitio web elegido proporciona dicha información.**

La finalidad del proyecto es educativa y de investigación, no comercial. **Idealista**, uno de los dos sitios web "raspados", es uno de los portales inmobiliarios de referencia en el contexto estatal y probablemente uno de los mayores portales de compra venta en España. Esta web pone al alcance del usuario multitud de información inmobiliaria para hacer más fácil al usuario su experiencia en la compraventa de inmuebles.

También se extraen datos del portal de datos financiero **Infobolsa**, que aporta datos oficiales de BME, Bolsas y Mercados Españoles. Seleccionamos este portal para descargar los datos financieros del IBEX35 porque la estructura de nombre de URL y su estructura HTML nos permite hacer peticiones efectivas anuales de datos de este índice.

2. **Definir un título para el dataset. Elegir un título que sea descriptivo.**

Llamaremos al dataset "Euribor, precios de la vivienda y cotizaciones Ibex35"

Autores:

Enrique J. Andrés Orera
Sergio Fernández Bertolín

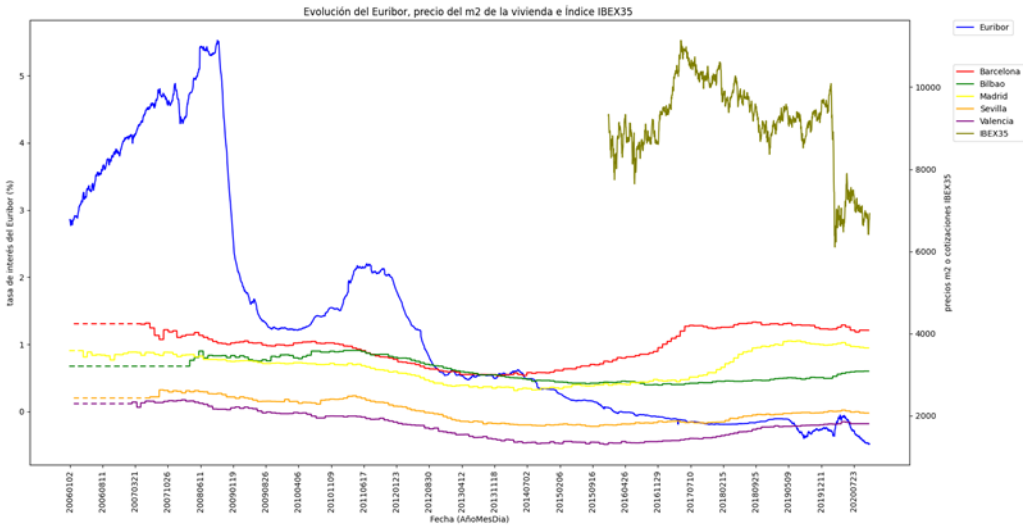
3. **Descripción del dataset. Desarrollar una descripción breve del conjunto de datos que se ha extraído (es necesario que esta descripción tenga sentido con el título elegido).**

Creación de un dataset con los valores diarios del índice del Euribor y los precios de venta medios de la vivienda/m^2 desde 2006 hasta el día actual. Adjuntamos también las cotizaciones al cierre diario de mercado del Ibex35 desde 2016, con fines educativos

Se obtienen las cotizaciones del Euribor y los precios de venta medios de la vivienda con un programa en Python mediante técnicas de webscraping contra el portal inmobiliario www.idealista.com

Las cotizaciones del Ibex35 se obtienen de www.infobolsa.com.

4. **Representación gráfica. Presentar una imagen o esquema que identifique el dataset visualmente**



Autores:

Enrique J. Andrés Orera
Sergio Fernández Bertolín

5. **Contenido. Explicar los campos que incluye el dataset, el periodo de tiempo de los datos y cómo se ha recogido.**

El Dataset generado tiene el siguiente formato:

Index	Dia	Euribor	precio_m2_Barcelona	precio_m2_Bilbao	precio_m2_Madrid	precio_m2_Sevilla	precio_m2_Valencia	IBEX35
2473	20160104	0.058	3313	2814	2743	1778	1342	9313.200
2474	20160105	0.059	3313	2814	2743	1778	1342	9335.200

Donde **Index** es un valor entero auto incremental, empezando en cero

Dia es el día en el que se ha registrado el valor, con el formato AAAAMMDD, donde las cuatro primeras cifras son el año, las dos siguientes el mes en formato numérico y las dos últimas el día del mes

Euribor es el valor del índice del Euribor registrado ese día, en formato decimal

precio_m2_Barcelona es el precio de venta medio por metro cuadrado en la ciudad de Barcelona

precio_m2_Bilbao es el precio de venta medio por metro cuadrado en la ciudad de Bilbao

precio_m2_Madrid es el precio de venta medio por metro cuadrado en la ciudad de Madrid

precio_m2_Sevilla es el precio de venta medio por metro cuadrado en la ciudad de Sevilla

precio_m2_Valencia es el precio de venta medio por metro cuadrado en la ciudad de Valencia

IBEX35 es la cotización diaria al cierre del indice Ibex 35

El periodo de tiempo de recogida es desde el 2 de enero de 2006 hasta el 30 de octubre de 2020 y se ha recogido mediante un raspado recurrente de las páginas

Autores:

Enrique J. Andrés Orera
Sergio Fernández Bertolín

de idealista con la información diaria y mensual del índice Euribor, y de los precios de venta medios/m^2. También se ha obtenido un raspado del índice Ibex 35 entre 1 de enero de 2016 y el día actual de la página www.infobolsa.com. En esta página no se proporcionan datos de antigüedad mayor a 5 años si la suscripción no es de pago y se ha utilizado una suscripción gratuita.

6. **Agradecimientos. Presentar al propietario del conjunto de datos. Es necesario incluir citas de investigación o análisis anteriores (si los hay).**

Idealista es uno de los portales inmobiliarios de referencia en el contexto estatal y probablemente uno de los mayores portales de compra venta en España.

Infobolsa es el portal de información financiera de BME Bolsas y Mercados Españoles.

Desconocemos la existencia de estudios previos.

7. **Inspiración. Explique por qué es interesante este conjunto de datos y qué preguntas se pretenden responder.**

En el momento actual, dentro de la mayor pandemia del siglo, hay una incertidumbre creciente en todos los aspectos financieros, de los que no queda exento el mercado inmobiliario.

Mediante esta recolección de datos se pretende obtener una herramienta que permita analizar la tendencia del principal índice inmobiliario en Europa, el Euribor, la tendencia en los precios inmobiliarios y la tendencia de la bolsa de valores, para poder realizar tanto análisis como predicciones.

8. **Licencia. Seleccione una de estas licencias para su dataset y explique el motivo de su selección: ○ Released Under CC0: Public Domain License ○ Released Under CC BY-NC-SA 4.0 License ○ Released Under CC BY-SA 4.0 License ○ Database released under Open Database License, individual contents under Database Contents License ○ Other (specified above) ○ Unknown License**

Hemos escogido la licencia **CC BY-NC-SA 4.0 License**, porque creemos que recoge adecuadamente el espíritu de la creación del dataset. Un dataset creado para un uso educativo, no comercial.

Autores:

Enrique J. Andrés Orera
Sergio Fernández Bertolín

Autores:

Enrique J. Andrés Orera
Sergio Fernández Bertolín

9. **Código. Adjuntar el código con el que se ha generado el dataset, preferiblemente en Python o, alternativamente, en R.**

```
# -*- coding: utf-8 -*-

"""

Script para extraer el euribor diario de la página web de Idealista

desde Enero de 2006 hasta el momento actual, así como datos del precio

de venta medio de la vivienda/m2

Extracción de datos de la cotización de cierre del Ibex35 de

www.infobolsa.com

"""


# Importación de librerías necesarias


import locale

import datetime

import random

from bs4 import BeautifulSoup as bsoup

import requests

import calendar

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import time
```

Autores:

Enrique J. Andrés Orera
Sergio Fernández Bertolín

```
from zipfile import ZipFile

from selenium import webdriver


# Necesario para calendar (configuración local a hora española)

locale.setlocale(locale.LC_ALL, 'es_ES.utf8')


# Creamos listas donde guardaremos los días y los valores

dias = []

valores = []


# Inicializaciones para interfaz con datos IBEX35

username_input = '//*[@id="login"]/ul[1]/li[1]/input'

password_input = '//*[@id="login"]/ul[1]/li[2]/input'

remember_input = '//*[@id="login"]/ul[1]/li[3]/input'

login_submit = '//*[@id="login"]/ul[1]/li[2]/a'

logout = '//*[@id="ifb-menu"]/div/ul/li[2]/a[2]'


# Lista de user agents

userAgents = [

    'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.113 Safari/537.36',

    'Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.90 Safari/537.36',
```

Autores:

Enrique J. Andrés Orera
Sergio Fernández Bertolín

'Mozilla/5.0 (Windows NT 5.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.90 Safari/537.36',

'Mozilla/5.0 (Windows NT 6.2; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.90 Safari/537.36',

'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/44.0.2403.157 Safari/537.36',

'Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/60.0.3112.113 Safari/537.36',

'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/57.0.2987.133 Safari/537.36',

'Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/57.0.2987.133 Safari/537.36',

'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/55.0.2883.87 Safari/537.36',

'Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/55.0.2883.87 Safari/537.36',

'Mozilla/4.0 (compatible; MSIE 9.0; Windows NT 6.1)',

'Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko',

'Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0)',

'Mozilla/5.0 (Windows NT 6.1; Trident/7.0; rv:11.0) like Gecko',

'Mozilla/5.0 (Windows NT 6.2; WOW64; Trident/7.0; rv:11.0) like Gecko',

'Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; rv:11.0) like Gecko',

'Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.0; Trident/5.0)',

'Mozilla/5.0 (Windows NT 6.3; WOW64; Trident/7.0; rv:11.0) like Gecko',

'Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0)',

'Mozilla/5.0 (Windows NT 6.1; Win64; x64; Trident/7.0; rv:11.0) like Gecko',

Autores:

Enrique J. Andrés Orera
Sergio Fernández Bertolín

```
    'Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; WOW64; Trident/6.0)',
    'Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; Trident/6.0)',
    'Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR
2.0.50727; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)'
]



# Función para extraer el euribor diario


def get_euribor(dias, valores):
    # Fecha actual
    now = datetime.datetime.now()


    anyo = int(now.strftime("%Y"))
    # Defino año final 2006, que es hasta donde tengo datos de precios
    anyo_final = 2006
    mes = int(now.strftime("%m"))


    controlmesactual = True


    # Para cada año
    for i_anyo in range(anyo, anyo_final-1, -1):
        # Para cada mes
```

Autores:

Enrique J. Andrés Orera
Sergio Fernández Bertolín

```
    for i_mes in range(mes, 0, -1):
        if controlmesactual:
            url = "https://www.idealista.com/news/euribor/mensual/mes-actual/"
            controlmesactual = False
        else:
            url = "https://www.idealista.com/news/euribor/mensual/%s-%d/" % (
            calendar.month_name[int(i_mes)], i_anyo)

        # Seleccionamos user agent aleatoriamente
        userAgent = random.choice(userAgents)

        # Cargamos cabeceras por defecto
        headers = requests.utils.default_headers()

        # Actualizamos cabeceras con el User-Agent aleatorio
        headers.update({'User-Agent': userAgent})

        # Espaciado entre peticiones (2 ó 3 segundos)
        sleep_secs = random.randrange(2, 4)
        time.sleep(sleep_secs)

        # Descargamos el sitio web de interés
        html = requests.get(url, headers=headers)
```

Autores:

Enrique J. Andrés Orera
Sergio Fernández Bertolín

```
            soup = bsoup(html.content)


            contador = 0


            for dato in soup.body.tbody.find_all('td'):
                contador = contador + 1
                if contador % 2 == 1:
                    # fecha
                    fecha = "%d%s%s" % (i_anyo, '{:02d}'.format(i_mes),
'{:02d}'.format(int(dato.string)))
                    # Añadimos la fecha a la lista
                    dias.append(fecha)
                    print(fecha)
                else:
                    # euribor
                    euribor = dato.string[:-1].replace(",", ".")
                    # Añadimos el euribor a la lista
                    valores.append(float(euribor))
                    print(float(euribor))
        mes = 12


    return euribor
```

Autores:

Enrique J. Andrés Orera
Sergio Fernández Bertolín

```
# Función para extraer el IBEX35 diario

def precios_ibex(dict_ibex, anio):

    url_anio = "https://www.infobolsa.es/cotizacion/historico-
ibex_35?startDate=%d0101&endDate=%d1231" % (anio, anio)


    # Opciones para el driver de Selenium

    options = webdriver.ChromeOptions()


    # Headless impide que el navegador Crhrome controlado por python se
muestre en pantalla

    options.add_argument('headless')


    # Ignoramos posibles errores

    options.add_argument('--ignore-certificate-errors')

    options.add_argument('--ignore-ssl-errors')


    # Elegimos el user-agent del pool de user-agents userAgents

    options.add_argument('user-agent=%s' % (random.choice(userAgents)))


    # Creamos objeto webdriver (selenium), que es el que realiza la petición con
las opciones anteriormente establecidas

    try:
```

Autores:

Enrique J. Andrés Orera
Sergio Fernández Bertolín

```
        web_driver = webdriver.Chrome(options=options)
    except:
        # Si no podemos crear el objeto webdriver es porque no tenemos el driver Chrome.
        # Lo descargamos y descomprimimos.
        print("Descargamos y descomprimimos driver Chrome")
        driver = requests.get('http://chromedriver.storage.googleapis.com/86.0.4240.22/chromedriver_win32.zip')

        with open('chrome_driver.zip', 'wb') as d:
            d.write(driver.content)

        with ZipFile('chrome_driver.zip', 'r') as zfile:
            try:
                zfile.extractall()
            except:
                print("Something else went wrong")
        # Una vez descargado y descomprimido el driver Chrome, creamos objeto webdriver (selenium), que es el que
        # realiza la petición con las opciones anteriormente establecidas.
        web_driver = webdriver.Chrome(options=options)

    # Espaciado entre peticiones (de 10 a 15 segundos)
```

Autores:

Enrique J. Andrés Orera
Sergio Fernández Bertolín

```
    sleep_secs = random.randrange(10, 15)

    time.sleep(sleep_secs)


    # Tiempo de espera del driver para asegurarse de que carga la página en su totalidad, 5 segundos

    seconds = 5

    web_driver.implicitly_wait(seconds)

    # Se abre una nueva petición a la página deseada

    web_driver.get('https://www.infobolsa.es/account/login')


web_driver.find_element_by_xpath(username_input).send_keys('sergiofbertolin@gmail.com')

    web_driver.find_element_by_xpath(password_input).send_keys('7y.zRt47ywsi')

    web_driver.find_element_by_xpath(remember_input).click()

    web_driver.find_element_by_xpath(login_submit).click()

    web_driver.get(url_anio)


    # Cambiamos al marco

    web_driver.switch_to.frame

    # Obtenemos el marco

    web_driver.page_source


    soup_tabla_ibex = bsoup(web_driver.page_source, "lxml")
```

Autores:

Enrique J. Andrés Orera
Sergio Fernández Bertolín

```
    anyomes = []
    ibex35 = []
    contador = 0
    for dato_tabla_ibex in soup_tabla_ibex.body.tbody.find_all('td'):
        contador += 1
        if contador % 6 == 1:
            print(dato_tabla_ibex.string)
            a = dato_tabla_ibex.string
            anyomes = "%s%s%s" % (a[6:10], a[3:5], a[0:2])
        elif contador % 6 == 2:
            print(dato_tabla_ibex.string)
            b = dato_tabla_ibex.string
            indice = b.replace(".", "")
            indice = indice.replace(",", ".")
            ibex35.append(indice)
        elif contador % 6 == 0:
            contador = 0
            if anyomes in dias:
                dict_ibex[anyomes] = indice

    # Cerramos driver
    web_driver.get('https://www.infobolsa.es/auth/bye')
    web_driver.close()
```

Autores:

Enrique J. Andrés Orera
Sergio Fernández Bertolín

```
    return (dict_ibex)



# Función para extraer los precios medios por m2 de una Ciudad.

# Se trata de un contenido dinámico.

# Utilizando la libreria Selenium, podremos acceder al marco donde se encuentra la tabla que se genera de forma

# dinámica


def precios_ciudad(dias, ciudad):

    ciudad_url_dict = {

        'Barcelona': 'https://www.idealista.com/sala-de-prensa/informes-precio-vivienda/venta/cataluna/barcelona-provincia/barcelona/historico/',

        'Bilbao': 'https://www.idealista.com/sala-de-prensa/informes-precio-vivienda/venta/euskadi/vizcaya/bilbao/historico/',

        'Madrid': 'https://www.idealista.com/sala-de-prensa/informes-precio-vivienda/venta/madrid-comunidad/madrid-provincia/madrid/historico/',

        'Sevilla': 'https://www.idealista.com/sala-de-prensa/informes-precio-vivienda/venta/andalucia/sevilla-provincia/sevilla/historico/',

        'Valencia': 'https://www.idealista.com/sala-de-prensa/informes-precio-vivienda/venta/comunitat-valenciana/valencia-valencia/valencia/historico/'}



    url_ciudad = ciudad_url_dict[ciudad]
```

Autores:

Enrique J. Andrés Orera
Sergio Fernández Bertolín

```
    # Opciones para el driver de Selenium
    options = webdriver.ChromeOptions()

    # Headless impide que el navegador Crhrome controlado por python se muestre en pantalla
    options.add_argument('headless')

    # Ignoramos posibles errores
    options.add_argument('--ignore-certificate-errors')
    options.add_argument('--ignore-ssl-errors')

    # Elegimos el user-agent del pool de user-agents userAgents
    options.add_argument('user-agent=%s' % (random.choice(userAgents)))

    # Creamos objeto webdriver (selenium), que es el que realiza la petición con las opciones anteriormente establecidas
    try:
        web_driver = webdriver.Chrome(options=options)
    except:
        # Si no podemos crear el objeto webdriver es porque no tenemos el driver Chrome.
        # Lo descargamos y descomprimimos.
        print("Descargamos y descomprimimos driver Chrome")
```

Autores:

Enrique J. Andrés Orera
Sergio Fernández Bertolín

```
        driver =
requests.get('http://chromedriver.storage.googleapis.com/86.0.4240.22/chrome
driver_win32.zip')


        with open('chrome_driver.zip', 'wb') as d:
            d.write(driver.content)


        with ZipFile('chrome_driver.zip', 'r') as zfile:
            try:
                zfile.extractall()
            except:
                print("Something else went wrong")
        # Una vez descargado y descomprimido el driver Chrome, creamos objeto
webdriver (selenium), que es el que
        # realiza la petición con las opciones anteriormente establecidas.
        web_driver = webdriver.Chrome(options=options)


    # Espaciado entre peticiones (de 10 a 15 segundos)
    sleep_secs = random.randrange(10, 16)
    time.sleep(sleep_secs)


    # Tiempo de espera del driver para asegurarse de que carga la página en su
totalidad, 5 segundos
    seconds = 5
```

Autores:

Enrique J. Andrés Orera
Sergio Fernández Bertolín

```
    web_driver.implicitly_wait(seconds)

    # Se abre una nueva petición a la página deseada

    web_driver.get(url_ciudad)


    # Buscamos el elemento 'iframe' que es el marco donde se encuentra la tabla
dinámica que queremos obtener.

    frame = web_driver.find_element_by_tag_name('iframe')

    # Cambiamos al marco

    web_driver.switch_to.frame

    # Obtenemos el marco

    web_driver.page_source


    soup_tabla_precios = bsoup(web_driver.page_source, "lxml")


    anyomes = []

    precio_m2 = []

    contador = 0

    for dato_tabla_precios in soup_tabla_precios.body.tbody.find_all('td'):

        contador += 1

        if contador % 5 == 1:

            a = dato_tabla_precios.string.split()

            for i in range(1, 13):

                if calendar.month_name[i] == a[0].lower():
```

Autores:

Enrique J. Andrés Orera
Sergio Fernández Bertolín

```
                anyomes.append("%s%s" % (a[1], '{:02d}'.format(i)))
        elif contador % 5 == 2:
            b = dato_tabla_precios.string.split()
            if b[0] == 'n.d.':
                precio_m2.append(np.NaN)
            else:
                precio = b[0].replace(".", "")
            precio_m2.append(precio)

    diccionario = dict(zip(anyomes, precio_m2))

    precio_m2_2 = []
    for i in range(len(dias)):
        try:
            precio_m2_2.append(np.int(diccionario[dias[i][0:6]]))
        except:
            precio_m2_2.append(np.NaN)

    # Cerramos driver
    web_driver.close()

    return precio_m2_2
```

Autores:

Enrique J. Andrés Orera
Sergio Fernández Bertolín

```
get_euribor(dias, valores)


start_period = 2020

end_period = 2015


# Inicializo un diccionario con NAs para todos los días

dict_ibex = {}

for i in range(len(dias)):

    dict_ibex[dias[i]] = np.NaN


for i in range(start_period, end_period, -1):

    dict_ibex = precios_ibex(dict_ibex, i)


ciudades = ['Barcelona', 'Bilbao', 'Madrid', 'Sevilla', 'Valencia']


# Creamos diccionario de listas con las listas días, valores,
precios_m2_Barcelona, precios_m2_Bilbao,

# precios_m2_Madrid, precios_m2_Sevilla, precios_m2_Valencia

euribor_dict = {'Dia': dias[::-1], 'Euribor': valores[::-1],

                'precio_m2_' + ciudades[0]: precios_ciudad(dias, ciudades[0])[::-1],

                'precio_m2_' + ciudades[1]: precios_ciudad(dias, ciudades[1])[::-1],

                'precio_m2_' + ciudades[2]: precios_ciudad(dias, ciudades[2])[::-1],
```

Autores:

Enrique J. Andrés Orera
Sergio Fernández Bertolín

```
            'precio_m2_' + ciudades[3]: precios_ciudad(dias, ciudades[3])[::-1],
            'precio_m2_' + ciudades[4]: precios_ciudad(dias, ciudades[4])[::-1],
            'IBEX35': list(dict_ibex.values())[::-1]}

# Creamos un DataFrame para almacenar el diccionario de listas
euribor_df = pd.DataFrame(euribor_dict)

# Almacenamos los resultados de nuestro dataset en un csv
euribor_df.to_csv('euribordiario.csv')

# Representamos gráficamente la evolución temporal del Euribor
print(euribor_df)

f, ax1 = plt.subplots()
ax1.plot(euribor_df.index, euribor_df.Euribor, color='blue', label='Euribor')
ax1.set(xlabel='Fecha (AñoMesDia)', ylabel='tasa de interés del Euribor (%)',
       title='Evolución del Euribor, precio del m2 de la vivienda e Índice IBEX35 ')
plt.xticks(np.arange(euribor_df.shape[0])[::150], euribor_df.Dia[::150],
rotation=90)
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0.)
# un segundo eje que comparte el eje x
ax2 = ax1.twinx()
ax2.set_ylabel('precios m2 o cotizaciones IBEX35')
```

Autores:

Enrique J. Andrés Orera
Sergio Fernández Bertolín

```
ax2.plot(euribor_df.index, euribor_df.precio_m2_Barcelona, color='red',
label='Barcelona')

ax2.plot(euribor_df.index, euribor_df.precio_m2_Bilbao, color='green',
label='Bilbao')

ax2.plot(euribor_df.index, euribor_df.precio_m2_Madrid, color='yellow',
label='Madrid')

ax2.plot(euribor_df.index, euribor_df.precio_m2_Sevilla, color='orange',
label='Sevilla')

ax2.plot(euribor_df.index, euribor_df.precio_m2_Valencia, color='purple',
label='Valencia')

ax2.plot(euribor_df.index, euribor_df.IBEX35, color='olive', label='IBEX35')

plt.legend(bbox_to_anchor=(1.05, 0.90), loc='upper left', borderaxespad=0.)

plt.show()

# Almacenamos la gráfica

f.savefig("euribor.png")
```

Autores:

Enrique J. Andrés Orera
Sergio Fernández Bertolín

10. Dataset. Publicación del dataset en formato CSV en Zenodo (obtención del DOI) con una breve descripción.

Creación de un dataset con los valores diarios del índice del Euribor y los precios de venta medios de la vivienda/m2 desde 2006 hasta el 9 de noviembre de 2020. Adjuntamos también las cotizaciones al cierre diario de mercado del Ibex35 desde 2016, con fines educativos.

```
https://doi.org/10.5281/zenodo.4264801
```

CONTRIBUCIONES	FIRMAS
Investigación previa	EJAO, SFB
Redacción de las respuestas	EJAO, SFB
Desarrollo del código	EJAO, SFB

Autores:

Enrique J. Andrés Orera
Sergio Fernández Bertolín