M2.851 · Tipología y ciclo de vida de los datos
Máster universitario de Ciencia de datos (*Data Science*)

Tipología y ciclo de vida de los datos

PRÁCTICA 1 : Web Scraping

Autores: Ignacio Fernández y Alba Rollano

Contexto

Bajo el contexto de la asignatura Tipología y Ciclo de vida de los Datos del Master en Ciencia de datos de la UOC, a lo largo del presente informe, se detallaran las tareas de *web scraping* realizadas sobre la web pisos (https://www.pisos.com/) con el objetivo de generar, en última instancia, un dataset con informacion detallada de los pisos en venta en las diferentes localidades españolas.

Para ello será necesario:

- Identificar los **datos relevantes** ,
- Recolección de datos a partir de **web crawling**
- Generación del dataset **pisos.csv** con toda la información a destacar.

Adicionalmente, toda la documentación quedará documentada en GitHub:House_Scrapping (https://github.com/arollano/House_Scrapping)

Motivacion

La presente crisis sanitaria ha acarreado un importante impacto no solo en la economía tanto nacional como mundial, sino también en los requisitos a las viviendas (ventanas, terrazas, espacio). Con el objetivo de evaluar ambos aspectos, surge la idea de comparar los pisos en venta a nivel nacional. Cuyos datos, permitirán responder a las siguientes preguntas, entre otras:

- ¿Cuál es el **precio medio por metro cuadrado** en las ciudades?
- ¿Varía respecto a localidades más pequeñas?
- ¿Cómo es el **precio vs. Superficie**?
- ¿Ha bajado?
- ¿A mayor numero de habitaciones mayor precio?

Dataset

1. Título

El dataset generado es **pisos.csv**

2. Descripción

El archivo **pisos.csv** contendrá información seleccionada de los pisos disponibles en venta en las diferentes localidades españolas en la web pisos (https://www.pisos.com/). Este archivo recoge las posibles propuestas que tendría un usuario al navegar en su interfaz:

3. Contenido

Dicho archivo contendrá para cada una de las entradas (pisos) la siguiente información seleccionada:

- **Summary**: Título del anuncio,
- **Description**: Descripción del piso,
- **Photo**: URL de la imagen principal,
- **Recommended**: Recomendado por la plataforma,
- **Price**: precio del piso,
- **Size**: tamaño,
- **Rooms**: número de habitaciones,
- **Price/m2**: precio por m2,
- **Bathrooms**: número de baños,
- **Num Photos**: número de imagenes disponibles,
- **Type**: tipo de vivienda**,
- **Region**: provincia

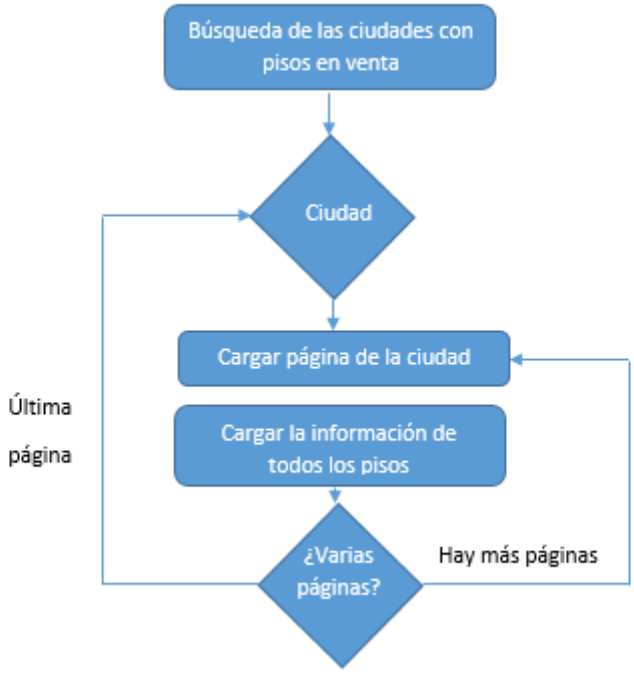
Es importante recalcar, que los datos aqui recogidos son los pisos activos en el momento de ejecución del script (*8/11/2020*)

summary	description	location	photo	recomendado	price	size	rooms	price/m2	bathrooms	Num Photos	type
Piso en Cerdanyola Nord	MATARÓ - CERDAÑOLA NORTE -. EN EXCLUSIVA PARA ...	Cerdanyola Nord (Mataró)	https://fotos5.imghs.net/nrd//516696/673628145...	Recomendado	139.000 €	96 m²	4.0	1.447 €/m²	2	16	pisos
Piso en Carrer de Numància	Pis en venda a Manresa, Zona Plaça Catalunya. ...	Valldaura-Plaça Catalunya (Manresa)	https://fotos1.imghs.net/nrd//500831/848582522...	Recomendado	145.000 €	128 m²	3.0	1.132 €/m²	2	15	pisos
Piso en Peramàs	MATARÓ - PERAMÁS -. Este bonito inmueble se en...	Peramàs (Mataró)	https://fotos1.imghs.net/nrd//516696/100086931...	Recomendado	116.000 €	64 m²	2.0	1.812 €/m²	1	9	pisos
Piso en Avinguda de Josep Tarradellas, cerca d...	Amplio y luminoso piso en la zona alta de Barc...	La Nova Esquerra de l'Eixample (Distrito Eixam...	https://fotos1.imghs.net/nrd//997263/583924549...	Recomendado	630.000 €	91 m²	2.0	6.923 €/m²	2	24	pisos
Casa en Castellbisbal	Bonita y acogedora casa de 80m², con gran saló...	Castellbisbal	https://fotos5.imghs.net/nrd//1005/536/1005_87...	Recomendado	234.900 €	183 m²	3.0	1.283 €/m²	2	36	pisos

4. Adquisición

La adquisición de los datos descritos anteriormente, conllevan un proceso previo de :

- Análisis del código web pisos (https://www.pisos.com/) para identificar el acceso a la misma
- Identificacion de los endpoint bloqueados por el archivo robot.txt (https://www.pisos.com/robots.txt)
- Obtencion de datos mediante *python* según el siguiente esquema:



- Carga en **pisos.csv**

No obstante para información más detallada, consultar el archivo Jupyter Notebooks de GitHub:House_Scrapping (https://github.com/arollano/House_Scrapping)

5. Agradecimientos

La información empleada en el estudio pertenece a la web pisos.com, un sitio web dedicado a la venta de pisos en España. Pisos.com nace en 2009 en Internet con el objetivo de ayudar a los demandantes de pisos a encontrar la casa de sus sueños y a los propietarios a encontrar inquilinos. Una breve introducción sobre su historia se puede encontrar en su sitio web: pisos (https://www.pisos.com/sobrepisos/). No se ha buscado ni utilizado ningún trabajo previo sobre esta web, para la realización de este trabajo.

6. Licencia

Código

In [ ]:

```
def parseHTML(url):
    '''
    '''
    # Download url
    site = requests.get(url)

    # Parse using BS
    soup = BeautifulSoup(site.content, 'html')

    return(soup)

def getPisos(soup):
    '''
    '''
    # Return pisos list given by label
    return(soup.findAll("div", {"class": "row clearfix"}))

def cleanAttribute(attribute):
    '''
    Function to remove \r\n and blanc spaces from the attributes

    INPUT: attribute
    '''
    attribute = attribute.replace('\r\n', '')
    return(attribute.strip())

def getTypeAndRegion(path):
    '''
    Function to get the region and piso type from a path

    INPUT: path
    RETURN:
    - piso_type: piso, estudio, apartamento...
    - region: the region the house belongs to
    '''
    # Clean path
    path = path.replace('/venta/', '')
    path = path.replace('/', '')

    # Get type and region
    piso_type = path[:path.find('-')]
    region = path[path.find('-')+1:]
    return(piso_type, region)


def getPisosAttributesFromPage(pisos_html, path):
    '''
    Scrap main info for each 'piso' scrapped

    INPUT:
    - pisos: Parsed html objetct with all div class='row clearfix'
    - region: region where the piso is located

    RETURN: pisos. Array containing a dict for each house for sale
    - summary
    - description
    - location
    - photo
    - recomendado
```

```
    - price
    - offer
    - size
    - rooms
    - price per m2
    - num photos
    '''
    pisos = []

    for dev in pisos_html:

        piso = {}

        # Get main attributes
        piso['summary'] = cleanAttribute(dev.h3.a.text)
        piso['description'] = cleanAttribute(dev.find('div', {'class':'description'}).t
ext)

        # Get location
        if (dev.find('div', {'class':'location'}) != None):
            piso['location'] = cleanAttribute(dev.find('div', {'class':'location'}).tex
t)
        else:
            piso['location'] = ''

        # Get photos
        if (dev.find('img')['src'] != '/Images/assets/blank1x1.png' and dev.find('img')
['src'] != None):
            piso['photo'] = dev.find('img')['src']
        elif (dev.find('img')['data-lazy'] == 'true'):
            piso['photo'] = dev.find('img')['data-lazy-img']

        # Get recomendado
        if (dev.find('div', {'class':'tag exclusivo'}) != None):
            piso['recomendado'] = cleanAttribute(dev.find('div', {'class':'tag exclusiv
o'}).text)
        else:
            piso['recomendado'] = ''

        # Get price
        piso['price'] = cleanAttribute(dev.find('div', {'class':'price'}).text)

        # Get size
        if (dev.find('div', {'class':'item'}) != None):
            piso['size'] = cleanAttribute(dev.find('div', {'class':'item'}).text)
        else:
            piso['size'] = ''

        # Get rooms if defined
        if (dev.find('div', {'class':'item', 'data-rooms':'true'}) != None):
            piso['rooms'] = cleanAttribute(dev.find('div', {'class':'item', 'data-room
s':'true'}).text)
        else:
            piso['rooms'] = ''

        piso['price/m2'] = ''

        # Get bathrooms
        if (dev.find('span', {'class': 'icon icoBath'}) != None):
            piso['bathrooms'] = cleanAttribute(dev.find('span', {'class': 'icon icoBat
h'}).parent.text)
```

```
        else:
            piso['bathrooms'] = ''

        # Get price per m2
        for item in dev.findAll('div', {'class':'item'}):
            if (item.text.find("€/m") != -1):
                piso['price/m2'] = cleanAttribute(item.text)

        # Get number of photos
        if (dev.find('div', {'class':'numPhotos'}) != None):
            piso['Num Photos'] = cleanAttribute(dev.find('div', {'class':'numPhotos'}).
text)
        else:
            piso['Num Photos'] = ''

        # Get region and type
        piso['type'], piso['region'] = getTypeAndRegion(path)

        # Append to outcome
        pisos.append(piso)

    # Return array
    return(pisos)
```

In [ ]:

```
def scanPath(domain, path):
    '''
    Given a path scan all pisos in the site as well as the next pages with the same sea
rch pattern

    INPUT: path. e.g. /venta/pisos-huelva/
    OUTPUT: object with all pisos found
    '''
    url = domain + path
    pisos = []

    # Parse site
    soup = parseHTML(url)

    # Scan pisos
    pisos_html = getPisos(soup)
    # Get pisos info
    pisos = getPisosAttributesFromPage(pisos_html, path)
    # Get next page
    next_page = soup.findAll('a', {'id':'lnkPagSig'})

    while len(next_page) != 0:
        for page in next_page:
            # Build  & parse url
            url = domain + page['href']
            soup = parseHTML(url)

            # Scan pisos
            pisos_html = getPisos(soup)
            # Get pisos info
            pisos = pisos + getPisosAttributesFromPage(pisos_html, path)

        # Get next page
        next_page = soup.findAll('a', {'id':'lnkPagSig'})

    return(pisos)
```

In [ ]:

```
def getCitiesToScan(domain, start_path):
    '''
    This function gets a start path and scan all cities offered in the browser

    INPUT: path to search
    OUTPUT: list of cities given by their paths to scan
    '''
    url = domain + start_path
    soup = parseHTML(url)

    # Find all cities in the browser
    cities = soup.findAll('span', {'class':'black-link'})

    # Extract link
    paths = []
    for city in cities:
        page = city.a
        if page != None:
            paths.append(page['href'])

    return(paths)
```

Las funciones detalladas anteriormente, quedan recogidas en el programa principal siguiente:

In [ ]:

```
domain = "https://www.pisos.com"
path = "/venta/pisos-barcelona/"

# Get all the cities from the pisos.com browser
cities = getCitiesToScan(domain, path)

list_cities = []

# Find all options in the brower for each page and build the cities list
# (depending on the city, the browser changes)
for city in cities:
    new_cities = getCitiesToScan(domain, city)
    for new_city in new_cities:
        # Append to the list if it's not there
        try:
            list_cities.index(new_city)
        except:
            list_cities.append(new_city)

print('Ciudades a buscar: ' + str(len(list_cities)))

pisos = []

# Get all pisos for sale for each city
for city in list_cities:
    pisos = pisos + scanPath(domain, city)
    print(len(pisos))

print('Búsqueda terminada')
```

Publicando, en última instancia, el archivo **pisos.csv**:

In [ ]:

```
import pandas as pd

pisos = pd.DataFrame(pisos)
pisos.head()

# Export to csv
pisos.to_csv('pisos.csv', encoding='utf-8')
```

Publicación

La base de datos **pisos.csv** está publicada en Zenodo en el siguiente enlace:
https://zenodo.org/record/4263693#.X6leV8hKiUl (https://zenodo.org/record/4263693#.X6leV8hKiUl)

Contribuciones

- **Investigación Previa** - Ignacio Fernández & Alba Rollano
- **Redacción de las respuestas** - Ignacio Fernández & Alba Rollano
- **Desarrollo código** - Ignacio Fernández & Alba Rollano

In [ ]:

```

```