Tipología y ciclo de vida de los datos

Práctica 1 (25% nota final)

Título: Web Scraping – Informes de calidad del aire en Andalucía

Alumnos:

Daniel Lugo Laguna

Pablo Mora Galindo

1. Contexto

De acuerdo a la Organización Mundial de la Salud, la contaminación del aire es un importante factor de riesgo para la salud. La salud cardiovascular y respiratoria de la población, tanto a corto como a largo plazo, mejora con niveles más bajos de contaminación del aire. **[1]**

Además de los problemas de salud anteriormente mencionados, la contaminación atmosférica también está estrechamente relacionada con el cambio climático, ya que parte de un mismo problema inicial: el modelo energético actual y el incremento en las emisiones de CO2. A esto se suma la emisión de otras partículas contaminantes como óxidos de nitrógeno (NO y NO_2), los óxidos de azufre (SO_2 y SO_3) o las partículas en suspensión. **[2]** Desde diferentes organismos internacionales, así como gobiernos nacionales se está trabajando en el control y reducción de estas emisiones por los graves afectos que acarrean.

Un paso fundamental en este aspecto, es disponer de información amplia y veraz sobre los valores de contaminantes en distintos lugares y momentos del tiempo. Esto permite establecer líneas de actuación dentro de una estrategia unificada para la toma de acciones preventivas y correctivas. Permite además la construcción de modelos de predicción, técnicas de *data mining* y otros análisis más profundos sobre el dato. Los gobiernos cada vez son más conscientes de esta necesidad de información y ponen a disposición del ciudadano estos datos a través de recursos web. Sin embargo, en muchas ocasiones esta información no se presenta de forma amigable para el análisis, sino que se publican como datos embebidos en web, dificultando su extracción, como es el presente caso de los informes diarios de calidad del aire, proporcionados por la Junta de Andalucía. **[3]**

2. Título del dataset

Como nombre del dataset se ha seleccionado: ***Evolución de la calidad del aire en Andalucía en el año 2020.***

3. Descripción del Dataset

El objetivo de este proyecto es, mediante técnicas de Web Scraping, obtener datos de calidad del aire para un período determinado de tiempo y para todas las provincias de Andalucía. De esta forma, esta información puede ser empleada en análisis posteriores sobre la evolución temporal y geográfica de la calidad atmosférica. Cada registro del dataset muestra por tanto la medición de una estación meteorológica situada en una ubicación específica y en un momento del tiempo (fecha y hora). Las mediciones consisten en la concentración de diversos contaminantes en el aire (CO, NO2, O3, PART, SO2, SH2), expresados en microgramos por metro cúbico ($\frac{\mu g}{m^3}$)

Se debe destacar que, con el código desarrollado, es posible obtener información de calidad del aire desde que existen registros en los informes publicados por la Junta de Andalucía (1 de enero de 1998). Sin embargo, dado que esto produciría un fichero muy extenso, para el presente proyecto se proporcionará un dataset con los registros existentes para el año 2020 y todas las provincias andaluzas. Mediante una variante del código original el usuario puede, si lo desea, obtener los datos históricos para un rango de tiempo personalizado.

4. Representación gráfica

En la Ilustración 1, se muestra un esquema general del proyecto desarrollado y presentado en este documento.

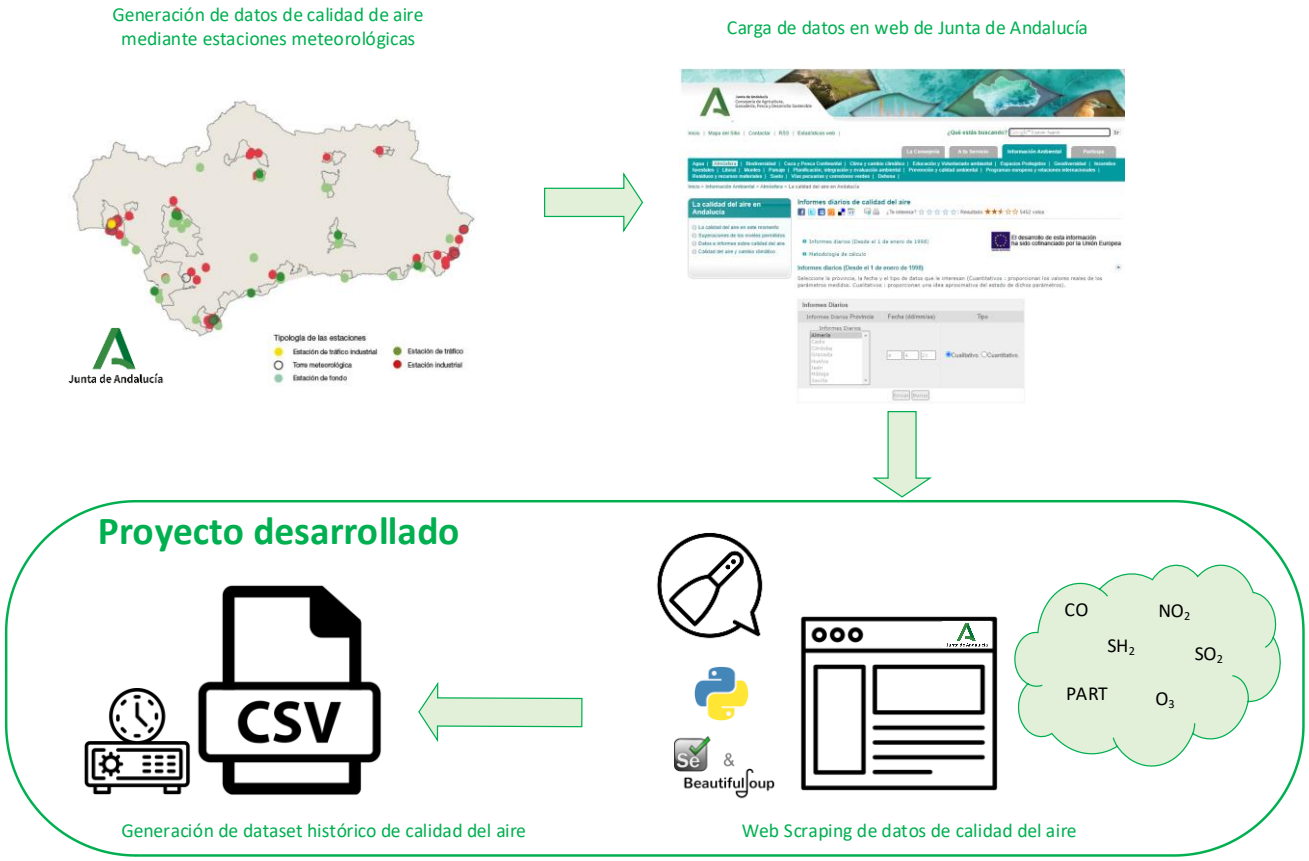

*Ilustración 1 Representación gráfica del proyecto (fuente: elaboración propia con iconos de **[4]** y **[5]**)*

5. Contenido

En la tabla 1, se muestra un diccionario de los campos incluidos en el dataset

Tabla 1 Diccionario de campos del Dataset generado (fuente: elaboración propia)

Nombre del campo	Descripción
index	Índice del registro medido para una ubicación, estación y día.
Provincia	Provincia donde se ha realizado la medición.
Municipio	Municipio donde se ha realizado la medición.
Dirección	Localización donde se sitúa la estación meteorológica de medición.
Estación	Designación/nombre de la estación meteorológica de medición.
FECHA-HORA	Fecha y hora a la que se realizó la medición de calidad atmosférica. Formato: DD/MM/AA-HH:mm.
CO	Concentración de **CO** en el aire medida en microgramos por metro cúbico ($\frac{\mu g}{m^3}$).
NO_2	Concentración de **NO_2** en el aire medida en microgramos por metro cúbico ($\frac{\mu g}{m^3}$).
O_3	Concentración de **O_3** en el aire medida en microgramos por metro cúbico ($\frac{\mu g}{m^3}$).
PART	Concentración de **partículas en suspensión** presentes en el aire, medidas en microgramos por metro cúbico ($\frac{\mu g}{m^3}$).
SO_2	Concentración de **SO_2** en el aire medida en microgramos por metro cúbico ($\frac{\mu g}{m^3}$).
SH_2	Concentración de **SH_2** en el aire medida en microgramos por metro cúbico ($\frac{\mu g}{m^3}$).

A continuación, se comentan una serie de particularidades específicas del dataset generado:

- La periodicidad de las mediciones depende de la estación específica que toma los datos. De forma general, la información suministrada por la página Web de la junta de Andalucía muestra registros cada 10 minutos.
- Se da el caso frecuente de valores perdidos para una estación en determinados. Estos valores pueden deberse a errores en la medición, fallos de Hardware, etc. En el dataset estos casos se han almacenado como NULL.

- No todas las estaciones miden todos los contaminantes de este estudio. En estos casos la estación tendrá todos los valores de este contaminante no medido a NULL.

6. Agradecimientos

La Junta de Andalucía, a través de la Consejería de Medio Ambiente y Ordenación del Territorio, es responsable de la información de calidad del aire Andaluza, objeto de este proyecto. Esta información se pone a disposición del ciudadano de forma abierta a través de recursos online. El desarrollo de esta herramienta Web, por parte de la Junta de Andalucía, ha sido cofinanciado por la Unión Europea. **[3]**

La contaminación atmosférica, así como las consecuencias que conlleva, tanto al medioambiente como a la salud de las personas, es un tema bastante relevante actualmente, con implicación de múltiples agentes gubernamentales, empresariales, políticos, científicos, etc. Por esta razón, existen informes y análisis publicados por distintas fuentes oficiales. El más relevante de ellos en el marco del proyecto actual es el informe titulado *Estrategia Andaluza de Calidad del Aire* **[6]**, publicado por la Junta de Andalucía y aprobado en el Boletín Oficial de la Junta de Andalucía (BOJA), a fecha de 28 de septiembre de 2020. **[7]**

Este documento realiza en primer lugar un diagnóstico actual de la calidad del aire subdividiendo Andalucía en diferentes regiones y proporcionando la evolución histórica de determinados valores promedios de distintos contaminantes. Así mismo, se comparan estas mediciones con los valores límite establecidos por la legislación vigente, con los valores objetivo planteados en las directivas europeas y con las guías de calidad del aire de la Organización Mundial de la Salud (OMS). A partir de los resultados obtenidos, se plantean una serie de objetivos estratégicos en la reducción de niveles de calidad del aire. Estos objetivos son posteriormente traducidos en diferentes niveles de obligatoriedad para la realización de los planes. Para alcanzar los objetivos se establecen una serie de medidas concretas en función de la zona y principales contaminantes. Finalmente se elabora una propuesta de evaluación y seguimiento mediante un sistema de indicadores de desempeño.

El Gobierno de España, a través del Ministerio para la Transición Ecológica y el Reto Demográfico, permite consultar el denominado Visor de Calidad del Aire. Esta herramienta web muestra la calidad del aire en España respecto a aquellos contaminantes con valores legislados de cara a proteger la salud de acuerdo al Real Decreto 102/2011. La información proporcionada por el visor es en su mayoría, de tipo cualitativo con un código de colores. De acuerdo a este código los verdes indican baja concentración de contaminantes, mientras que los rojos indican altas concentraciones. Existen diferentes tonos intermedios del amarillo al naranja para indicar concentraciones medias. El visor permite personalizar la vista introduciendo o eliminando diferentes contaminantes a la interfaz, modificar la agrupación temporal, entre otras funciones. En la ilustración 2, se puede observar un ejemplo de esta interfaz.

Existen otras fuentes no gubernamentales que realizan análisis críticos sobre los datos oficiales publicados por las distintas Comunidades Autónomas (entre estos, los informes diarios de Calidad del Aire de Andalucía) como es el caso de Ecologistas en Acción, en su Informe titulado: *La calidad del aire en el Estado español durante 2019* **[9].** En este documento se proponen una serie de recomendaciones a criterio de la organización para la mejora de la calidad atmosférica.

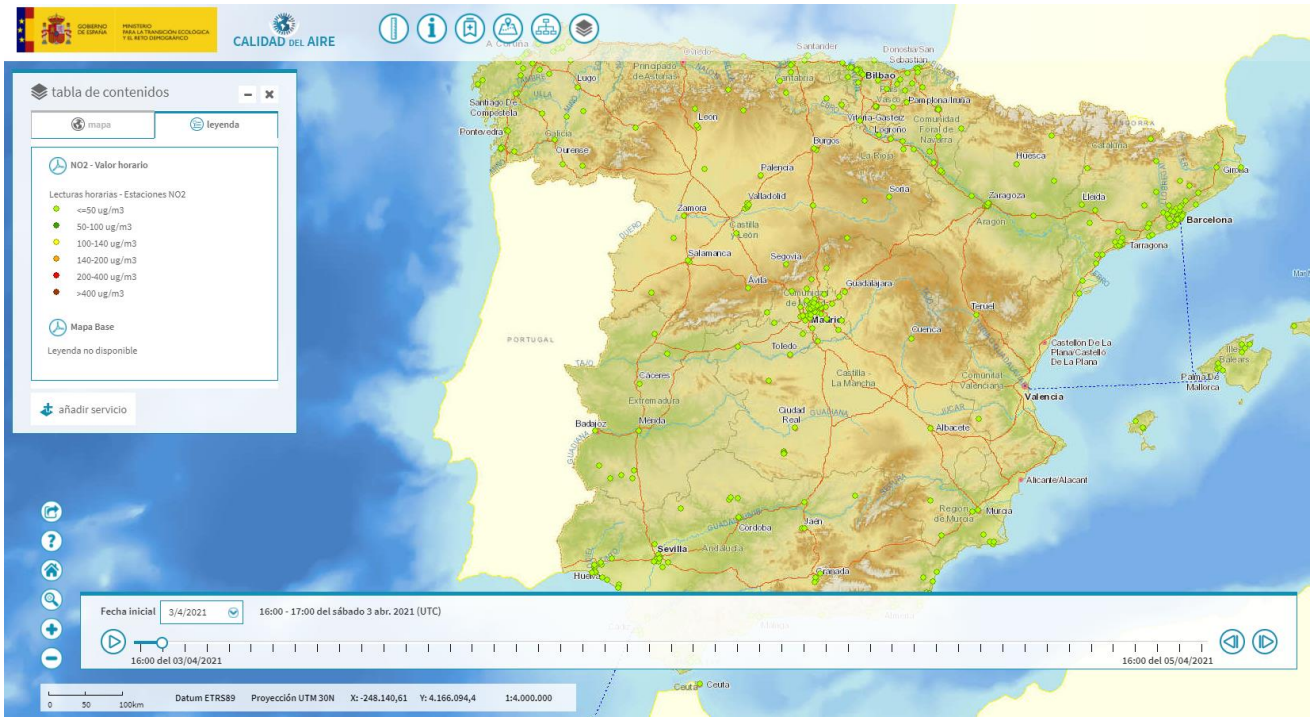

*Ilustración 2. Interfaz del Visor de Calidad del aire del Gobierno de España (fuente: sig.mapama.gob.es **[8]**)*

Se ha realizado una búsqueda de proyectos similares al presente, a través de diferentes repositorios de código abierto, principalmente GitHub. De los analizados, el más parecido al presente proyecto es el introducido en **[10]**, donde, mediante lenguaje R, se propone la publicación de una librería (denominada aire) que contiene funciones creadas expresamente para extraer los datos publicados en los informes diarios de calidad del aire de la Junta de Andalucía, con el objetivo de representar gráficamente los valores de contaminación. La librería prepara los datos para que puedan ser utilizados directamente con la librería de R *openair.*

Otros proyectos de carácter abierto como **[11]** y **[12]** se centran en extraer datos de calidad del aire de los sitios webs oficiales de otras Comunidades Autónomas (en este caso la Comunidad de Madrid), sin embargo, la labor de extracción en estos proyectos se reduce a obtener los datos de un fichero plano de texto tipo .txt.

7. Inspiración

A pesar de que el análisis de calidad del aire es un tema relevante para el gobierno Andaluz (con diferentes informes y herramientas publicados), la disponibilidad del dato en su máxima granularidad (esto es, el acceso a los datos brutos de medición de las estaciones) no es sencillo de obtener. Los datos están publicados en la página web de la Junta de Andalucía con una estructura poco amigable. Además, los resultados se presentan segmentados por día y por provincia.

Si un usuario deseara descargar manualmente los datos de un año completo para todas las provincias, debería visitar 365 x 8 = 2920 páginas. Esto sin contar que cada estación aparece en una tabla diferente en la Web con un formato de tabla complejo. Como ejemplo, en la ilustración 3 se muestra un fragmento de los datos para una estación meteorológica y día concreto tal como aparecen en el recurso web original.

Provincia	SEVILLA
Municipio	SEVILLA
Estacion	RANILLA
Direccion	Avd. ANDALUCIA

FECHA-HORA	SO2	NO2	CO
04/04/21-00:10	6	15	666
04/04/21-00:20	6	12	623
04/04/21-00:30	6	10	630
04/04/21-00:40	6	9	637
04/04/21-00:50	6	14	654
04/04/21-01:00	6	15	652
04/04/21-01:10	6	11	638
04/04/21-01:20	6	9	637
04/04/21-01:30	6	8	640
04/04/21-01:40	6	8	625
04/04/21-01:50	6	8	622
04/04/21-02:00	6	8	614
04/04/21-02:10	6	7	613
04/04/21-02:20	6	8	615
04/04/21-02:30	6	7	625
04/04/21-02:40	6	7	633
04/04/21-02:50	6	6	614
04/04/21-03:00	6	4	620
04/04/21-03:10	6	5	629
04/04/21-03:20	6	7	629
04/04/21-03:30	6	6	636

*Ilustración 3 Fragmento de los datos del informe diario de calidad del aire de Andalucía (fuente: Junta de Andalucía **[3]**)*

Este proyecto pretende facilitar esta tarea de extracción de los datos brutos al usuario final, de forma que, introduciendo el período de tiempo deseado en la función proporcionada, pueda extraer de forma rápida la información, evitando estas descargas manuales. Así mismo, se proporciona ya preparado un dataset con los datos completos para el año 2020.

El repositorio contiene dos versiones del código. Una de ellas devuelve el dataset completo de 2020. La segunda versión permite modificar los parámetros de entrada respecto a la provincia y el intervalo de tiempo requerido. Esto permite al usuario descargar únicamente los datos que desee en lugar de toda la información.

La aplicación de buenas prácticas a la hora de aplicar Web Scraping, resulta muy importante para no saturar las webs desde las que se recupera la información. Para este proyecto, se ha revisado en primer lugar el fichero robots.txt de la Web de la Junta de Andalucía, situado en la ruta: https://www.juntadeandalucia.es/robots.txt. El fichero muestra algunas prohibiciones expresas para cierto contenido, pero ninguno relacionado con el presente proyecto. En la ilustración 4 se muestra un fragmento de este registro. Por otra parte, se ha añadido al código un retraso de unos segundos entre cada petición a la página web. Además, se incluye una comprobación sobre que la petición no recibe de respuesta un error 404, esto indica problemas en la web y se pararía el proceso.

```
user-agent:*
Disallow: /icms/
Disallow: /institucional/
Disallow: /buscar.html
Disallow: /boja/buscador/search.do?
Disallow: /educacion/apl/consultabolsas/
Disallow: /empleo/www
Disallow: /GuiaFyS
Disallow: /iaapinstitutodeadministracionpublica/institutodeadministracionpublica/servlet/descarga
Disallow: /iaapinstitutodeadministracionpublica/servlet/descarga
Disallow: /institutodeadministracionpublica/institutodeadministracionpublica/servlet/descarga
Disallow: /institutodeadministracionpublica/moodle/mod/forum
Disallow: /institutodeadministracionpublica/servlet/descarga
Disallow: /organismos/culturaypatrimoniohistorico/iaph/areas/formacion-difusion/programa-formacion/calendario-actividades-formativas/formulario
Disallow: /organismos/fomentoinfraestructurasyordenaciondelterritorio/servicios/atencion-ciudadania/paginas/bolsa-unica-comun
Disallow: /PJA
Disallow: /rcja/sio
Disallow: /SP/
Disallow: /xwiki/bin/view/MADEJA/
Disallow: /export/drupaljda/ACTA%20PROVISIONAL%20DOCENCIA.pdf
Disallow: /export/drupaljda/ACTA%20SELEC%20DEFINITIVA%20DOCENCIA%2041-004%20.pdf
Disallow: /export/drupaljda/almeria agosto
Disallow: /export/drupaljda/ALMERIA_
Disallow: /export/drupaljda/almeria_
Disallow: /export/drupaljda/cadiz agosto
Disallow: /export/drupaljda/CADIZ_
Disallow: /export/drupaljda/cadiz_
Disallow: /export/drupaljda/cordoba agosto
Disallow: /export/drupaljda/CORDOBA_
Disallow: /export/drupaljda/cordoba_
Disallow: /export/drupaljda/granada agosto
Disallow: /export/drupaljda/GRANADA_
Disallow: /export/drupaljda/granada_
Disallow: /export/drupaljda/huelva agosto
Disallow: /export/drupaljda/HUELVA_
Disallow: /export/drupaljda/huelva_
Disallow: /export/drupaljda/jaen agosto
Disallow: /export/drupaljda/JAEN_
Disallow: /export/drupaljda/jaen_
Disallow: /export/drupaljda/malaga agosto
Disallow: /export/drupaljda/MALAGA_
Disallow: /export/drupaljda/malaga_
Disallow: /export/drupaljda/malaga-enero
Disallow: /export/drupaljda/sevilla agosto
Disallow: /export/drupaljda/SEVILLA_
Disallow: /export/drupaljda/sevilla_
Disallow: /export/videos/
Disallow: /boja/app
```

Ilustración 4 Fragmento del fichero robots.txt de la web de la Junta de Andalucía (fuente: juntadeandalucia.es)

Las aplicaciones de este dataset de histórico de mediciones en bruto pueden ser muy relevantes para diferentes tipos de usuarios finales. Puede ser un instrumento muy útil para la comunidad científica de cara a realizar análisis más profundos y detallados sobre los datos. Algunas de las preguntas que podrían responderse son las siguientes:

- *¿Existe una correlación entre la presencia de tipos concretos de contaminantes?*
- *¿Hay estacionalidad en la presencia de los contaminantes en el aire a lo largo del año?*
- *¿Cuáles son las horas pico de contaminación al día? ¿Han variado a lo largo del tiempo?*
- *¿Es comparable la evolución de la contaminación entre provincias?*
- *¿Se puede cuantificar el efecto del desarrollo urbano en la calidad del aire?*
- *¿Es posible generar un modelo predictivo de alta precisión sobre la concentración diaria de contaminación?*
- *¿Qué provincia ha empeorado más en promedio en calidad del aire en el último año?*

Estas son solo algunas de las preguntas que podrían resultar de interés en futuras investigaciones. Sin embargo, con las herramientas actuales (el visor y los informes mencionados en el apartado anterior **[6-9]**), la información que el usuario puede obtener de forma rápida sobre la contaminación atmosférica en Andalucía (sin recurrir al Web Scraping o a copias manuales masivas) es mayoritariamente de tipo cualitativo, con indicadores de tipo semáforo. Esto impide realizar los análisis en profundidad requeridos para responder a las preguntas planteadas.

El proyecto introducido en **[10],** referente a la librería aire, resulta de gran interés en este caso, ya que el objetivo es similar a este proyecto. Sin embargo, el repositorio de GitHub no presenta actividad desde 2016, por lo que la librería aire, generada en código R, ha podido quedar obsoleta. Este proyecto se ha desarrollado en Python con librerías de Web Scraping ampliamente utilizadas (Beautiful Soup y Selenium), lo que facilita una ampliación y soporte del código a corto y medio plazo.

Por otra parte, los datos de salida (principalmente fecha) de las funciones de la librería aire ya aparecen transformados para trabajar con openair. En el presente proyecto, los datos se extraen "as-is" en bruto, sin modificaciones. Se proporciona además en este proyecto un dataset de ejemplo con los datos para el año 2020, de forma que el usuario final pueda conocer la estructura de los datos de antemano.

8. Licencia

Como licencia para el código desarrollado en este proyecto, se ha seleccionado **"MIT License".** Se ha optado por ella por la flexibilidad que proporciona. **MIT License** está clasificada como una licencia de tipo *Permisive,* lo cual implica la no obligación a publicar como "open source" el código fuente de otros desarrolladores, en el caso de que usen parte del código de este proyecto para otro proyecto diferente, lo cual ocurre con las licencias de tipo *Copyleft* (cualquier trabajo derivado se debe distribuir con la licencia original, como en la licencia GPL por ejemplo). Los usuarios de código con licencia MIT pueden modificar la fuente original y distribuirlo como un proyecto independiente.

Con la selección de esta licencia se busca no limitar ni forzar a los desarrolladores a publicar su código original. Se pretende por tanto que el código sea lo más aprovechable posible en futuros proyectos y desarrollos.

9. Código

M2.851_PRAC1_dlugol_pmoraga.py

Código para la extracción completa del dataset para todas las provincias y el año 2020.

```
import requests
from bs4 import BeautifulSoup
import pandas as pd
import numpy as np
import math, time
from itertools import zip_longest
```

```
from selenium import webdriver
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from datetime import datetime, timedelta

URL_main =
'http://www.juntadeandalucia.es/medioambiente/site/portalweb/menuitem.7e1cf46ddf59bb227a9eb
e205510e1ca/?vgnextoid=7e612e07c3dc4010VgnVCM1000000624e50aRCRD&vgnextchannel=910f230af77e4
310VgnVCM1000001325e50aRCRD#apartadoce217adf12be4010VgnVCM1000000624e50a____'

#Función creada para reorganizar la lista de medidas dividiéndola en bloques en función de
la longitud de su cabecera
def grouper(n, it_list, padValue=None):
    """ n -> nºcolumnas por fila
        it_list -> lista sobre la que iterar
        padValue -> valor por defecto
    """
    return zip_longest(*[iter(it_list)]*n, fillvalue=padValue)

# Opciones de navegación
options =  webdriver.ChromeOptions()
options.add_argument('--start-maximized')
options.add_argument('--disable-extensions')

# Al descargarnos dicho ejecutable, debemos añadir la ruta del directorio webdrivers a
nuestra variable de entorno del sistema PATH
driver_path = 'C:/Users/Pablo/Documents/webdrivers/chromedriver.exe'

driver = webdriver.Chrome(driver_path, options=options)

# Inicializamos el navegador
driver.get(URL_main)

# Para obtener el código de estado de la URL usamos un javascript y un objeto
XMLHttpRequest
js_status = '''
let callback = arguments[0];
let xhr = new XMLHttpRequest();
xhr.open('GET',
'http://www.juntadeandalucia.es/medioambiente/site/portalweb/menuitem.7e1cf46ddf59bb227a9eb
e205510e1ca/?vgnextoid=7e612e07c3dc4010VgnVCM1000000624e50aRCRD&vgnextchannel=910f230af77e4
310VgnVCM1000001325e50aRCRD#apartadoce217adf12be4010VgnVCM1000000624e50a____', true);
```

```
xhr.onload = function () {
    if (this.readyState === 4) {
        callback(this.status);
    }
};
xhr.onerror = function () {
    callback('error');
};
xhr.send(null);
'''

# LLamamos al siguiente método de la clase WebDriver para ejecutar el código javascript
anterior dentro del navegador
status_code = driver.execute_async_script(js_status)

# Comprobamos que la petición nos devuelve un Status Code = 200
if status_code == 200:
    #Inicialización del esquema del dataframe final
    df_final = pd.DataFrame({ 'FECHA-HORA': [],'SO2': [],'PART': [], 'NO2': [], 'CO': [],
'O3': [], 'SH2': [],
                              'Provincia': [], 'Municipio': [], 'Estacion': [], 'Direccion':
[]})

    # Obtenemos los 365 días del año 2020 en una lista para su uso posterior
    ini_2020 = datetime(2020,1,1)
    fin_2020 = datetime(2020,12,31)
    l_fechas_2020 = [(ini_2020 + timedelta(days=d)).strftime("%Y-%m-%d")
                        for d in range((fin_2020 - ini_2020).days + 1)]

    # Lista de provincias
    l_prov_id = ['al', 'ca', 'co', 'gr', 'hu', 'ja', 'ma', 'se']
    for prov_id in l_prov_id:
        # Seleccionamos la provincia correspondiente
        WebDriverWait(driver, 5)\
            .until(EC.element_to_be_clickable((By.XPATH,
"//option[@value='{}']".format(prov_id))))\
            .click()

        # Seleccionamos siempre el tipo "Cuantitativo" para los informes
        WebDriverWait(driver, 5)\
            .until(EC.element_to_be_clickable((By.XPATH, "//input[@value='n' and
@name='TIPO']")))\
            .click()
```

```
        # Recorremos dicha lista para ir insertando los valores en los campos DIA/MES/AÑO y
poder realizar todas las consultas
        for dia in l_fechas_2020:
            # Borramos los valores por defecto dados a Fecha(dd/mm/aa)
            driver.find_element_by_name("DIA").clear()
            driver.find_element_by_name("MES").clear()
            driver.find_element_by_name("ANO").clear()

            aa = dia.split('-')[0][-2:]
            mm = dia.split('-')[1]
            dd = dia.split('-')[2]

            # Asignamos a DIA/MES/AÑO sus valores correspondientes en cada iteración
            WebDriverWait(driver, 5)\
                .until(EC.element_to_be_clickable((By.NAME,"DIA")))\
                .send_keys(dd)

            WebDriverWait(driver, 5)\
                .until(EC.element_to_be_clickable((By.NAME,"MES")))\
                .send_keys(mm)

            WebDriverWait(driver, 5)\
                .until(EC.element_to_be_clickable((By.NAME,"ANO")))\
                .send_keys(aa)

            # Accedemos a la página con los informes de la provincia, fecha y tipo
correspondiente
            WebDriverWait(driver, 5)\
                .until(EC.element_to_be_clickable((By.XPATH,
"/html/body/div[1]/div[6]/div[2]/div/div/div[3]/div[2]/form/table/tbody/tr[3]/td/input[1]")
))\
                .send_keys(Keys.CONTROL + Keys.RETURN)

            # Actuamos en PÁGINA de INFORMES
            soup = BeautifulSoup(driver.page_source, 'html.parser')
            tables = soup.findAll("table")
            l_province = list()
            l_municipio = list()
            l_station = list()
            l_dir = list()
            #Iteramos por todas las tablas:
            #Índice para iterar por las filas del dataframe de localización
            j = 0
            for table in tables:
```

```
                tabla_measures = list()
                #Se ha utilizado la palabra clave provincia para filtrar
                if table.text.startswith('Provincia'):
                    indice_prov = table.text.index('Provincia')
                    indice_mun = table.text.index('Municipio')
                    indice_est = table.text.index('Estacion')
                    indice_dir = table.text.index('Direccion')
                    l_province.append(table.text[indice_prov+9:indice_mun])
                    l_municipio.append(table.text[indice_mun+9:indice_est])
                    l_station.append(table.text[indice_est+8:indice_dir])
                    l_dir.append(table.text[indice_dir+9:])

                    #Variable de control que indica que tabla se está recorriendo en dicha
iteración, de localización (1) o de mediciones (0)
                    #Esto permite ir incrementando el índice para la tabla paramétrica de
provincias
                    ind_tabla_prov = 1

                elif not table.text.startswith('RED DE VIGILANCIA'):
                    l_cab = list()
                    table_Cabecera = table.find_all('td', {'class': 'CabTabla'})
                    for cab in table_Cabecera:
                        l_cab.append(cab.text)

                    measures_tr = table.find_all('tr')
                    measures = list()
                    for tr in measures_tr:
                        measures_td = tr.find_all('td')
                        for td in measures_td:
                            if td.text not in ('FECHA-HORA', 'SO2', 'PART', 'NO2', 'CO',
'O3', 'SH2') and not td.text.startswith('\nNota:'):
                                measures.append(td.text)

                    if len(l_cab) != 0:
                        tabla_measures = list([l_cab])
                        tabla_measures+=list(grouper(len(l_cab), measures,
math.ceil(len(measures)/len(l_cab))))
                        df_measures =
pd.DataFrame(tabla_measures[1:],columns=tabla_measures[0])

                    df_loc = pd.DataFrame({'Provincia': l_province, 'Municipio':
l_municipio, 'Estacion': l_station, 'Direccion': l_dir})
```

```
                    # Recuperamos todos los elementos cuya clase se denomine 'CabTabla', ya
que estos representan las cabeceras de las tablas de medidas.
                    # (en cada tabla de medida hay tantos 'CabTabla' como columnas haya).
Los almacenamos en una lista que recorremos y cuando se detecte
                    # el inicio de una cabecera (es decir, el texto "FECHA-HORA" es el
primero que aparece) incrementamos el contador num_tablas_medidas en uno.
                    # Con esto obtenemos el número total de tablas de medidas que hay en
cada página de informes.
                    lista_tablas_medidas = driver.find_elements_by_class_name('CabTabla')
                    num_tablas_medidas = 0
                    for indice in lista_tablas_medidas:
                        if indice.text == "FECHA-HORA":
                            num_tablas_medidas = num_tablas_medidas + 1

                    # Si el índice de la lista no ha llegado al nº máximo de tablas de
medidas o no es la primera tabla de provincia, se crea un
                    # dataframe intermedio con la posición de la paramétrica
correspondiente. A continuación, se replica la fila recogida tantas
                    # veces como filas tenga el dataframe de medidas intermedio del paso
anterior.
                    if l_province and j < num_tablas_medidas:
                        df_loc_iter = pd.DataFrame(np.repeat(pd.DataFrame({'Provincia':
l_province[j],
                                                           'Municipio': l_municipio[j],
                                                           'Estacion': l_station[j],
                                                           'Direccion':
l_dir[j]},index=[0]).values,len(df_measures),axis=0),columns = ['Provincia','Municipio',

'Estacion','Direccion'])
                        # Se concatenan los dos df intermedios de medidas y localizaciones
para la presente iteración
                        # Se añaden las filas al dataframe final con toda la información
                        df_iter = pd.concat((df_measures,df_loc_iter), axis = 1)
                        df_final = df_final.append(df_iter,sort=True)
                        # Si se ha recorrido tabla de provincia en el paso anterior, se
incrementa +1 el indicador
                        if ind_tabla_prov == 1:
                            j = j + 1
                        # Variable de control de tabla de provincia. Dado que se ha
recorrido una tabla de medidas, se pasa el indicador a 0
                        ind_tabla_prov = 0

            # Retrocedemos a la página principal
            driver.back()
```

```
    # Reordenamos columnas y quitamos indice
    df_final = df_final[['Provincia', 'Municipio','Direccion', 'Estacion','FECHA-
HORA','CO','NO2','O3','PART','SO2','SH2']].reset_index(drop=True)
    # Añadimos nuevo índice
    df_final = df_final.reset_index()

    # Generamos fichero AirQualityAndalusia2020.csv
    df_final.to_csv('AirQualityAndalusia2020.csv', index = False,
header=True,encoding='utf-8-sig',sep=',')
    driver.close()
else:
    print("Status code %d" % status_code)
```

M2.851_PRAC1_dlugol_pmoraga_param.py

Código para la extracción del dataset para provincia y período de tiempo personalizado por el usuario.

```
import requests
from bs4 import BeautifulSoup
import pandas as pd
import numpy as np
import math, time
from itertools import zip_longest
from selenium import webdriver
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
import sys
from selenium.webdriver.support.select import Select

URL_main =
'http://www.juntadeandalucia.es/medioambiente/site/portalweb/menuitem.7e1cf46ddf59bb227a9eb
e205510e1ca/?vgnextoid=7e612e07c3dc4010VgnVCM1000000624e50aRCRD&vgnextchannel=910f230af77e4
310VgnVCM1000001325e50aRCRD#apartadoce217adf12be4010VgnVCM1000000624e50a____'

#Función creada para reorganizar la lista de medidas dividiéndola en bloques en función de
la longitud de su cabecera
def grouper(n, it_list, padValue=None):
    """ n -> nºcolumnas por fila
        it_list -> lista sobre la que iterar
        padValue -> valor por defecto
```

```
    """
    return zip_longest(*[iter(it_list)]*n, fillvalue=padValue)

# Opciones de navegación
options =  webdriver.ChromeOptions()
options.add_argument('--start-maximized')
options.add_argument('--disable-extensions')

# Al descargarnos dicho ejecutable, debemos añadir la ruta del directorio webdrivers a
nuestra variable de entorno del sistema PATH
driver_path = 'C:/Users/Pablo/Documents/webdrivers/chromedriver.exe'

driver = webdriver.Chrome(driver_path, options=options)

# Inicializamos el navegador
driver.get(URL_main)

# Para obtener el código de estado de la URL usamos un javascript y un objeto
XMLHttpRequest
js_status = '''
let callback = arguments[0];
let xhr = new XMLHttpRequest();
xhr.open('GET',
'http://www.juntadeandalucia.es/medioambiente/site/portalweb/menuitem.7e1cf46ddf59bb227a9eb
e205510e1ca/?vgnextoid=7e612e07c3dc4010VgnVCM1000000624e50aRCRD&vgnextchannel=910f230af77e4
310VgnVCM1000001325e50aRCRD#apartadoce217adf12be4010VgnVCM1000000624e50a____', true);
xhr.onload = function () {
    if (this.readyState === 4) {
        callback(this.status);
    }
};
xhr.onerror = function () {
    callback('error');
};
xhr.send(null);
'''

# LLamamos al siguiente método de la clase WebDriver para ejecutar el código javascript
anterior dentro del navegador
status_code = driver.execute_async_script(js_status)

# Comprobamos que la petición nos devuelve un Status Code = 200
if status_code == 200:
    #Inicialización del esquema del dataframe final
```

```
    df_final = pd.DataFrame({ 'FECHA-HORA': [],'SO2': [],'PART': [], 'NO2': [], 'CO': [],
'O3': [], 'SH2': [],
                              'Provincia': [], 'Municipio': [], 'Estacion': [], 'Direccion':
[]})

    # Recuperamos los parámetros de entrada: provincia y fecha (YYYY-MM-DD)
    if len(sys.argv) == 3:
        prov_name = sys.argv[1]
        fecha = sys.argv[2]
        aa = fecha.split('-')[0][-2:]
        mm = fecha.split('-')[1]
        dd = fecha.split('-')[2]
    else:
        print('El número de parámetros de entrada introducido es ERRÓNEO. Deben ser
PROVINCIA y FECHA.')
        exit()

    # Recuperamos todos los valores del scroll select de PROVINCIA
    select_provincia = Select(WebDriverWait(driver,
5).until(EC.element_to_be_clickable((By.XPATH, "//select[@name='PROVINCIA']"))))
    # Seleccionamos la provincia correspondiente
    select_provincia.select_by_visible_text(prov_name)

    # Seleccionamos siempre el tipo "Cuantitativo" para los informes
    WebDriverWait(driver, 5)\
        .until(EC.element_to_be_clickable((By.XPATH, "//input[@value='n' and
@name='TIPO']")))\
        .click()

    # Borramos los valores por defecto dados a Fecha(dd/mm/aa)
    driver.find_element_by_name("DIA").clear()
    driver.find_element_by_name("MES").clear()
    driver.find_element_by_name("ANO").clear()

    # Insertamos los valores en los campos DIA/MES/AÑO
    WebDriverWait(driver, 5)\
        .until(EC.element_to_be_clickable((By.NAME,"DIA")))\
        .send_keys(dd)

    WebDriverWait(driver, 5)\
        .until(EC.element_to_be_clickable((By.NAME,"MES")))\
        .send_keys(mm)

    WebDriverWait(driver, 5)\
```

```
        .until(EC.element_to_be_clickable((By.NAME,"ANO")))\
        .send_keys(aa)

    # Accedemos a la página con los informes de la provincia, fecha y tipo correspondiente
    WebDriverWait(driver, 5)\
        .until(EC.element_to_be_clickable((By.XPATH,
"/html/body/div[1]/div[6]/div[2]/div/div/div[3]/div[2]/form/table/tbody/tr[3]/td/input[1]")
))\
        .send_keys(Keys.CONTROL + Keys.RETURN)

    # Actuamos en PÁGINA de INFORMES
    soup = BeautifulSoup(driver.page_source, 'html.parser')
    tables = soup.findAll("table")
    l_province = list()
    l_municipio = list()
    l_station = list()
    l_dir = list()
    #Iteramos por todas las tablas:
    #Índice para iterar por las filas del dataframe de localización
    j = 0
    for table in tables:
        tabla_measures = list()
        #Se ha utilizado la palabra clave provincia para filtrar
        if table.text.startswith('Provincia'):
            indice_prov = table.text.index('Provincia')
            indice_mun = table.text.index('Municipio')
            indice_est = table.text.index('Estacion')
            indice_dir = table.text.index('Direccion')
            l_province.append(table.text[indice_prov+9:indice_mun])
            l_municipio.append(table.text[indice_mun+9:indice_est])
            l_station.append(table.text[indice_est+8:indice_dir])
            l_dir.append(table.text[indice_dir+9:])

            #Variable de control que indica que tabla se está recorriendo en dicha
iteración, de localización (1) o de mediciones (0)
            #Esto permite ir incrementando el índice para la tabla paramétrica de
provincias
            ind_tabla_prov = 1

        elif not table.text.startswith('RED DE VIGILANCIA'):
            l_cab = list()
            table_Cabecera = table.find_all('td', {'class': 'CabTabla'})
            for cab in table_Cabecera:
                l_cab.append(cab.text)
```

```
            measures_tr = table.find_all('tr')
            measures = list()
            for tr in measures_tr:
                measures_td = tr.find_all('td')
                for td in measures_td:
                    if td.text not in ('FECHA-HORA', 'SO2', 'PART', 'NO2', 'CO', 'O3',
'SH2') and not td.text.startswith('\nNota:'):
                        measures.append(td.text)

            if len(l_cab) != 0:
                tabla_measures = list([l_cab])
                tabla_measures+=list(grouper(len(l_cab), measures,
math.ceil(len(measures)/len(l_cab))))
                df_measures = pd.DataFrame(tabla_measures[1:],columns=tabla_measures[0])

            df_loc = pd.DataFrame({'Provincia': l_province, 'Municipio': l_municipio,
'Estacion': l_station, 'Direccion': l_dir})

            # Recuperamos todos los elementos cuya clase se denomine 'CabTabla', ya que
estos representan las cabeceras de las tablas de medidas.
            # (en cada tabla de medida hay tantos 'CabTabla' como columnas haya). Los
almacenamos en una lista que recorremos y cuando se detecte
            # el inicio de una cabecera (es decir, el texto "FECHA-HORA" es el primero que
aparece) incrementamos el contador num_tablas_medidas en uno.
            # Con esto obtenemos el número total de tablas de medidas que hay en cada
página de informes.
            lista_tablas_medidas = driver.find_elements_by_class_name('CabTabla')
            num_tablas_medidas = 0
            for indice in lista_tablas_medidas:
                if indice.text == "FECHA-HORA":
                    num_tablas_medidas = num_tablas_medidas + 1

            # Si el índice de la lista no ha llegado al nº máximo de tablas de medidas o no
es la primera tabla de provincia, se crea un
            # dataframe intermedio con la posición de la paramétrica correspondiente. A
continuación, se replica la fila recogida tantas
            # veces como filas tenga el dataframe de medidas intermedio del paso anterior.
            if l_province and j < num_tablas_medidas:
                df_loc_iter = pd.DataFrame(np.repeat(pd.DataFrame({'Provincia':
l_province[j],
                                                   'Municipio': l_municipio[j],
                                                   'Estacion': l_station[j],
```

```
                                                'Direccion':
l_dir[j]},index=[0]).values,len(df_measures),axis=0),columns = ['Provincia','Municipio',

'Estacion','Direccion'])
                # Se concatenan los dos df intermedios de medidas y localizaciones para la
presente iteración
                # Se añaden las filas al dataframe final con toda la información
                df_iter = pd.concat((df_measures,df_loc_iter), axis = 1)
                df_final = df_final.append(df_iter,sort=True)
                # Si se ha recorrido tabla de provincia en el paso anterior, se incrementa
+1 el indicador
                if ind_tabla_prov == 1:
                    j = j + 1
                # Variable de control de tabla de provincia. Dado que se ha recorrido una
tabla de medidas, se pasa el indicador a 0
                ind_tabla_prov = 0

    # Retrocedemos a la página principal
    driver.back()

    # Reordenamos columnas y quitamos indice
    df_final = df_final[['Provincia', 'Municipio','Direccion', 'Estacion','FECHA-
HORA','CO','NO2','O3','PART','SO2','SH2']].reset_index(drop=True)
    # Añadimos nuevo índice
    df_final = df_final.reset_index()

    # Generamos fichero AirQualityAndalusia.csv
    df_final.to_csv('AirQualityAndalusia_{}_{}.csv'.format(prov_name, fecha), index =
False, header=True,encoding='utf-8-sig',sep=',')
    driver.close()
else:
    print("Status code %d" % status_code)
```

10. Dataset

El DOI del dataset en Zenodo es:

https://zenodo.org/record/4681703#.YHSYxOj7SUk

Breve Descripción: Dataset histórico de calidad del aire en Andalucía. Los datos han sido tomados de la página Web oficial de la Junta de Andalucía, en el apartado de informes SIVA y mediciones de tipo cuantitativo. Contiene valores de contaminantes medidos en microgramos por metro cúbico tomadas durante el año 2020 por las estaciones meteorológicas presentes en las provincias de

Andalucía (España). El dataset se ha publicado bajo licencia MIT, al igual que el código que permite su cálculo y obtención.

Contribuciones individuales al proyecto

Contribuciones	Firma
Investigación previa	**D.L.L., P.M.G.**
Redacción de las respuestas	**D.L.L., P.M.G.**
Desarrollo código	**D.L.L., P.M.G.**

Referencias

1. Calidad del aire ambiente (exterior) y salud https://www.who.int/es/news-room/fact-sheets/detail/ambient-(outdoor)-air-quality-and-health (último acceso: 1 de Abril, 2021).

2. Relación entre el cambio climático y la contaminación del aire https://www.sostenibilidad.com/cambio-climatico/relacion-cambio-climatico-contaminacion-del-aire/ (último acceso: 1 de Abril, 2021).

3. Informes diarios de calidad del aire - Consejería de Medio Ambiente y Ordenación del Territorio. Junta de Andalucía http://www.juntadeandalucia.es/medioambiente/site/portalweb/menuitem.7e1cf46ddf59bb227a9ebe205510e1ca/?vgnextoid=7e612e07c3dc4010VgnVCM1000000624e50aRCRD&vgnextchannel=910f230af77e4310VgnVCM1000001325e50aRCRD (último acceso: 1 de Abril, 2021).

4. Shin, Y.; Seoul, K. [Web] BeautifulSoup와 표현식 https://yganalyst.github.io/web/crawling_4/ (último acceso: Apr 5, 2021).

5. Noun Project: Free Icons & Stock Photos for Everything https://thenounproject.com/ (accessed Apr 5, 2021).

6. Estrategia Andaluza de Calidad del Aire - Junta de Andalucía - Consejería de Agricultura, Ganadería, Pesca y Desarrollo Sostenible Estrategia Andaluza de Calidad del Aire; 2020. Disponible en: http://www.juntadeandalucia.es/medioambiente/portal_web/web/temas_ambientales/atmosfera/estrategia_andaluza_ca/eaca_definitiva.pdf

7. Estrategia Andaluza de Calidad del Aire – Instituto Nacional de Administración pública http://laadministracionaldia.inap.es/noticia.asp?id=1202870 (último acceso: Apr 5, 2021).

8. Calidad del Aire https://sig.mapama.gob.es/calidad-aire/ (último acceso: Apr 5, 2021).

9. La calidad del aire en el Estado español durante 2019 Ecologistas en Acción; 2020 https://www.ecologistasenaccion.org/wp-content/uploads/2020/06/informe-calidad-aire-2019.pdf (último acceso: Apr 5, 2021).

10. SevillaR/aire https://github.com/SevillaR/aire (último acceso: Apr 5, 2021).

11. chucheria/CalidadAire https://github.com/chucheria/CalidadAire

12. UlisesGascon/Aire-Madrid https://github.com/UlisesGascon/Aire-Madrid

Recursos

- Subirats, L., Calvo, M. (2018). Web Scraping. Editorial UOC.
- Masip, D. El lenguaje Python. Editorial UOC.
- Lawson, R. (2015). Web Scraping with Python. Packt Publishing Ltd. Chapter 2. Scraping the Data.
- Simon Munzert, Christian Rubba, Peter Meißner, Dominic Nyhuis. (2015). Automated Data Collection with R: A Practical Guide to Web Scraping and Text Mining. John Wiley & Sons.
- Tutorial de Github https://guides.github.com/activities/hello-world.