Práctica 1 (25% nota final)

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

El contexto en el que se ha recolectado la información es la tienda online de Mercadona a través de su propia web utilizando la librería Selenium de Python. Mercadona es una de las principales cadenas de supermercados en España y ha desarrollado una web que permite acceder a información sobre sus productos, precios, ubicaciones de tiendas, entre otros datos relevantes.

El sitio web elegido proporciona información sobre los productos, precios y existencias de Mercadona, lo que es valioso para aquellos que quieren comparar precios, conocer la disponibilidad de los productos o simplemente realizar un seguimiento de los cambios en los precios de los productos a lo largo del tiempo.

La dirección del sitio web es https://www.mercadona.es.

2. **Título.** Definir un título conciso y que sea descriptivo para el *dataset*.

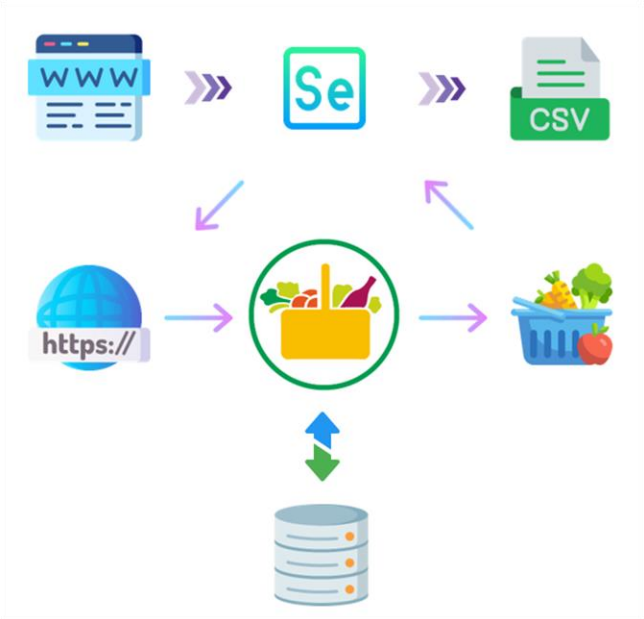
"Productos de Mercadona: información detallada, precios, marcas y categorías".

3. **Descripción del dataset.** Desarrollar una breve descripción del conjunto de datos que se ha extraído. Es necesario que esta descripción sea coherente con el título elegido.

El conjunto de datos extraído se titula "Productos de Mercadona" y contiene información sobre los productos que se venden en la tienda online de Mercadona. La información incluye el nombre del producto, su marca, su categoría, su precio, su descripción y su disponibilidad. Además, también se incluyen datos sobre el tamaño del producto y su imagen.

*El **dataset** se ha obtenido a través de la propia tienda web de Mercadona y contiene información actualizada hasta la fecha en que se realizó la extracción. Este conjunto de datos puede resultar útil para realizar análisis de mercado y/o comparar los precios y características de los productos vendidos por Mercadona.*

4. **Representación gráfica.** Dibujar un esquema o diagrama que refleje visualmente el dataset y el proyecto elegido.



5. **Contenido.** Explicar los campos que se incluyen en el dataset y el período de tiempo al que pertenecen los datos.

El dataset obtenido a través de la API de Mercadona contiene información sobre los productos de su tienda en línea. Los campos que se incluyen son los siguientes:

- *Grupo del que forma parte el producto.*
- *Categoría a la que pertenece el producto.*
- *Nombre del producto.*
- *Precio del producto.*
- *Cantidad de unidades que componen el producto.*
- *Unidad de medida (Si el producto se mide en kg, l, ml….)*
- *Precio por unidad.*

El período de tiempo al que pertenecen los datos no está especificado en la web, ya que los productos pueden haber sido agregados en diferentes momentos.

Sin embargo, se puede asumir que los datos son actuales y están actualizados constantemente, ya que provienen de la tienda en línea en funcionamiento de Mercadona.

6. **Propietario.** Presentar al propietario del conjunto de datos. Es necesario incluir citas de análisis anteriores o, en su defecto, justificar esta búsqueda con análisis similares. Indicar qué pasos se han seguido para actuar de acuerdo con los principios éticos y legales en el contexto del proyecto elegido.

En este caso, el propietario de los datos es la empresa Mercadona S.A., propietaria de la tienda online de donde se han extraído los datos a través de la web de la tienda. No se han encontrado análisis anteriores de este conjunto de datos en particular, ya que es exclusivo de la tienda online de Mercadona.

En cuanto a los principios éticos y legales, se han seguido los siguientes pasos:

1. *Limitar la cantidad de datos extraídos y almacenados, para evitar la sobrecarga del sitio web o la afectación a la privacidad de los usuarios.*
2. *Utilizar los datos sólo para fines de análisis y no para fines comerciales o ilegales.*
3. *Asegurar el anonimato en todo momento de datos personales o sensibles que puedan ser encontrados durante el proceso.*
4. *Respetar los derechos de autor y propiedad intelectual de Mercadona S.A. al citar adecuadamente la fuente en caso de ser usados estos datos por cualquier análisis o publicación posterior.*

Hay que recalcar nuevamente que, siempre se ha actuado de una manera licita para garantizar que se respeten los derechos de los propietarios de los datos y se actúe de forma ética y legal en todo momento.

7. **Inspiración.** Explicar por qué puede ser interesante este conjunto de datos y qué preguntas se pretenden responder con ellos. Es necesario comparar con los análisis anteriores o análisis similares presentados en el apartado 6.

Este conjunto de datos puede ser interesante para varias personas y/o empresas como, por ejemplo:

- *Investigadores y analistas de mercado: pueden utilizar esta información para analizar las tendencias en la industria de los supermercados, comparar precios y productos entre diferentes cadenas, identificar patrones de consumo, etc.*

- *Competidores de Mercadona: pueden utilizar esta información para analizar la estrategia de precios y productos de la empresa, identificar oportunidades de negocio, etc.*

Las preguntas que se pueden responder con estos datos son varias, por ejemplo:

- *¿Cuáles son los precios promedio de los productos en diferentes categorías (alimentos, productos de limpieza, etc.)?*
- *¿Cómo han variado los precios de ciertos productos con el tiempo?*
- *¿Cuáles son los productos que tienen mayor fluctuación de precio?*
- *¿Existen patrones de consumo en función del día de la semana o del mes?*
- *¿Cómo se comparan los precios y productos de Mercadona con los de otras cadenas de supermercados?*

Todas estas preguntas se pueden responder mediante análisis de estadísticas descriptivas, gráficas, comparaciones con otros conjuntos de datos, entre otros.

8. **Licencia.** Seleccionar una licencia adecuada para el dataset resultante y justificar el motivo de su elección.

La elección de la licencia en este caso puede ser muy variada dependiendo de la intención y propósito final del proyecto.

Para nuestra practica en concreto hemos decidido escoger la licencia CC0 que es una licencia de dominio público que permite la máxima libertad en el uso, distribución y modificación del dataset sin restricciones, incluso para fines comerciales.

Las razones para decidirnos sobre este tipo de licencia es la de permitir que otros investigadores, desarrolladores o la comunidad en general utilicen y mejoren el dataset sin restricciones.

9. **Código.** Código implementado para la obtención del dataset, preferiblemente en Python o, alternativamente, en R.

https://github.com/JosemiGT/Rastrea-Mercado

A continuación, se muestran los fragmentos del código más representativos del proceso de web scraping.

.../source/main.py

```
from scraper.MercadonaScraper import MercadonaScraper
from repository.CSVProductRepository import CSVProductRepository
import time
from datetime import datetime

OUTPUT_FILE = "datos_mercadona.csv"

if __name__ == "__main__":

    print("Iniciando Mercadona Scraper")

    today = datetime.today()
    date_string = today.strftime('%Y%m%d')

    # Creamos una instancia de MercadonaScraper() y especificamos el código postal
    mercadona_scraper = MercadonaScraper(46001)
    product_repository = CSVProductRepository(
        date_string + "_" + OUTPUT_FILE)

    product_repository.insert_headers()

    # Completamos el código postal y aceptamos las cookies
    mercadona_scraper.complete_postal_code()
    mercadona_scraper.accept_cookies()
    time.sleep(2)

    # Navegamos a la página de categorías
    mercadona_scraper.navigate_to_categories()

    # Obtenemos una lista de todos los grupos de categorías disponibles
    categories_groups = mercadona_scraper.get_all_groups_categories()

    there_are_next_category = True
    there_are_next_group = True

    # Recorremos cada uno de los grupos de categorías
    for group in categories_groups:

        # Navegamos a la siguiente página de categorías del grupo actual
        mercadona_scraper.navigate_to_next_group_categories(group)

        # Mientras haya más categorías disponibles en la página actual…
        while(there_are_next_category):
            # Guardamos los productos encontrados en la página en el archivo de salida
            product_repository.save_products(mercadona_scraper.get_all_product_page())
            # Navegamos a la siguiente página de categorías
            there_are_next_category = mercadona_scraper.navigate_to_next_categories()

        there_are_next_category = True

    # Cerramos el navegador
    mercadona_scraper.close_the_browser()
```

…/source/scraper/MercadonaScraper.py

```
from selenium import webdriver
from selenium.common.exceptions import NoSuchElementException
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.remote.webelement import WebElement
from selenium.webdriver.common.keys import Keys
import time
from models.product import Product

class MercadonaScraper():

    def __init__(self, postal_code:int):

        self.url:str = "https://tienda.mercadona.es/"
        self.delay_time = 5
        self.postal_code:int = postal_code
        self.driver = webdriver.Firefox()
        self.driver.implicitly_wait(30)
        self.driver.maximize_window()

        # Obtener el User-Agent utilizando JavaScript
        user_agent = self.driver.execute_script("return navigator.userAgent")
        print('El User-Agent es:', user_agent)

        self.driver.get(self.url)
        self.wait_for_page_to_load()

    # Método que define el código postal
    def complete_postal_code(self):

        postal_code_box = self.driver.find_element(By.NAME, 'postalCode')
        postal_code_box.send_keys(str(self.postal_code))

        postal_code_buttom = self.driver.find_element(
               By.XPATH, '//button[@data-test="postal-code-checker-button"]')
        postal_code_buttom.click()

        time.sleep(1)

    # Método para aceptar las cookies
    def accept_cookies(self):

        cookies_buttom = self.driver.find_element(
               By.XPATH, '//button[contains(text(), "Aceptar todas")]')
        cookies_buttom.click()

        time.sleep(1)

    # Método para navegar por la sección de categorías de productos
    def navigate_to_categories(self):

        categories_section = self.driver.find_element(
               By.XPATH, '//a[@href="/categories"]')
        categories_section.click()

        self.wait_for_page_to_load()
```

```
# Método para navegar por las subcategorías
def navigate_to_next_categories(self)-> bool:

    next_subcategories = self.driver.find_elements(
           By.XPATH, '//button[contains(@class, "next-subcategory")]')
    if(next_subcategories == None
       or len(next_subcategories) == 0):
        return False

    next_subcategories[0].click()
    self.wait_for_page_to_load()

    return True

# Método que devuelve todos los grupos de categorías de productos
def get_all_groups_categories(self):

    return self.driver.find_elements(
           By.XPATH, '//li[contains(@class, "category-menu__item")]')

# Método para mostrar más categorías dentro de un grupo
def navigate_to_next_group_categories(self, category_group):

    button = category_group.find_element(By.TAG_NAME, 'button')
    button.click()
    self.wait_for_page_to_load()

# Método que devuelve todos los productos de una página
def get_all_product_page(self):

    time.sleep(1)

    products:list[Product] = []
    set_products_button = self.driver.find_elements(
           By.XPATH, '//button[@data-test="open-product-detail"]')

    for button in set_products_button:
        try:
            products.append(self.get_product(button))

        except NoSuchElementException:
            time.sleep(5)
            continue

    return products

# Método que devuelve un producto
def get_product(self, button:WebElement) -> Product:

    try:
        button.click()
        time.sleep(1)
        product = self.get_product_information()

    except Exception as ex:
        print("Exception: {0}".format(ex))
        self.try_close_retries_windows()
        close_product_button = self.driver.find_element(
           By.XPATH, '//button[@data-test="modal-close-button"]')
```

```
            close_product_button.click()
            product = self.get_product(button)


        close_product_button = self.driver.find_element(
               By.XPATH, '//button[@data-test="modal-close-button"]')

        close_product_button.click()

        return product

    # Método para obtener información de un producto
    def get_product_information(self) -> Product:

        group_name = ""
        group_name_elements = self.driver.find_elements(
               By.XPATH, '//span[@class="subhead1-r"]')

        if group_name_elements and len(group_name_elements) > 0:
            group_name = group_name_elements[0].text

        group_category = ""
        group_name_categories = self.driver.find_elements(
               By.XPATH, '//span[@class="subhead1-sb"]')

        if group_name_categories and len(group_name_categories) > 0:
            group_category = group_name_categories[0].text

        product_name = self.driver.find_element(
               By.XPATH, '//h1[@class="title2-r private-product-detail__description"
                          and @tabindex="0"]').text
        price_web_elements = self.driver.find_elements(
               By.XPATH, '//p[@class="product-price__unit-price large-b"]')

        if price_web_elements and len(price_web_elements) > 0:
            product_price = price_web_elements[0].text
        else:
            product_price = self.driver.find_element(
               By.XPATH, '//p[@class="product-price__unit-price large-b
                         product-price__unit-price--discount"]').text

        product_amount = self.driver.find_element(
               By.XPATH, '//p[@class="product-price__extra-price title1-r"]').text

        div_product_information_text = self.driver.find_element(
               By.XPATH, '//div[@class="product-format product-format__size"
                          and @tabindex="0"]').text.split('|')

        product_unit = div_product_information_text[0]
        product_price_by_unit = div_product_information_text[1]

        return Product(
            group_name,
            group_category,
            product_name,
            product_price,
            product_amount,
            product_unit,
            product_price_by_unit)
```

```
# Método para intentar cerrar ventanas emergentes de intentos fallidos
def try_close_retries_windows(self):

    if(self.is_contains_retries_windows()):
        self.delay_time += 1
        print("Retry with delay {}".format(self.delay_time))
        time.sleep(self.delay_time)

# Método para comprobar si existe una ventana emergente de intentos fallidos
def is_contains_retries_windows(self):

    button = self.driver.find_elements(
           By.XPATH, '//button[contains(text(), "Entendido")]')

    if(button and len(button) > 0):
        print("Va a clickar botón de entendido")
        button[0].click()
        print("Se ha clickado")
        time.sleep(1)
        return True

    return False

# Método para esperar a que se cargue la página
def wait_for_page_to_load(self):
    WebDriverWait(self.driver, 10).until(
           EC.presence_of_element_located((By.TAG_NAME, 'body')))

# Método para cerrar el navegador
def close_the_browser(self):
    self.driver.quit()
```

10. **Dataset.** Publicar el dataset obtenido en formato CSV en *Zenodo*, incluyendo una breve descripción de este. Obtener y adjuntar el enlace del DOI del dataset (https://doi.org/…). El dataset también deberá incluirse en la carpeta **/dataset** del repositorio.

https://doi.org/10.5281/zenodo.7864711

11. **Vídeo**. Realizar un breve vídeo explicativo de la práctica **(máximo 10 minutos)**, que deberá contar con la participación de los dos integrantes del grupo. En el vídeo se deberá realizar una presentación del proyecto, destacando los puntos más relevantes, tanto de las respuestas a los apartados como del código utilizado para extraer los datos. Indicar el enlace del vídeo.

https://drive.google.com/file/d/1t9pVGXzJNzeS9a2oXhKAN6pbT0CDFsNP/view?usp=share_link

.

Contribuciones	Firma
• Investigación previa • Redacción de las respuestas • Desarrollo del código • Participación en el vídeo	*Salvador Pulido Sánchez* *José Miguel Gamarro Tornay*