FACULTAD 50 AÑOS DE INGENIERÍA UNaM
JIDeTEV 2024 ISSN 2591-4219
XIV Jornadas de Investigación, Desarrollo Tecnológico, Extensión, Vinculación y Muestra de la Producción

Diseño de metodología de un algoritmo de web scraping para la adquisición de datos de clima a partir de BeautifulSoup

Briant A. Gauna [a, b, *], Fabricio Pasinato [b], Nancy B. Ganz [a, c]

[a] *Facultad de Ingeniería, Universidad Nacional de Misiones (UNaM), Oberá, Misiones, Argentina.*
[b] *Universidad del Gran Rosario (UGR) – U. Acad. Diseño y Tecnologías, Rosario, Santa Fe, Argentina.*
[c] *Facultad de Ciencias Exactas, Químicas y Naturales – Módulo Apóstoles, Universidad Nacional de Misiones (UNaM), Apóstoles, Misiones, Argentina*
E-mails: br.gauna@fio.unam.edu.ar, fpasinato6@gmail.com, nancyganz@fceqyn.unam.edu.ar

Resumen

Este artículo explora la relevancia de los datos en la era digital, enfatizando cómo el internet actúa como una vasta fuente de información que los investigadores deben navegar para extraer datos útiles. Se centra en la técnica de web scraping, utilizando Python y la librería BeautifulSoup, como herramienta esencial para la adquisición de datos estructurados a partir de la web. El ensayo propone una metodología basada en la teoría de la "V" de Big Data, que abarca desde la selección de fuentes de información hasta la limpieza y almacenamiento de datos en formatos como CSV. Se aplicó esta metodología en un ejemplo práctico, utilizando el sitio Wunder Underground para extraer datos meteorológicos de Posadas durante dos años.

Los resultados subrayan la efectividad del web scraping, aunque requiere un esfuerzo considerable en la limpieza de los datos extraídos. Las conclusiones indican que es posible establecer criterios claros para la selección de webs adecuadas para scraping y que la sistematización de este proceso es factible. Finalmente, se sugiere explorar nuevos criterios de clasificación de webs y comparar diferentes herramientas y técnicas en futuros estudios para optimizar la adquisición y procesamiento de datos a partir de web scraping.

Palabras Clave *– Algoritmos, Big Data, Web Scraping, Beautiful Soup, Minería de Datos, Ciencia de Datos*

1 Introducción

“Los datos son el nuevo petróleo”, o “el mundo está lleno de datos” son frases muy habituales de escuchar. Llegan a ser parte de la jerga popular. Sin embargo, estos datos están distribuidos y esparcidos caóticamente en registros, planillas de cálculo, sistemas de gestión de bases de datos, entre otros soportes.

El lugar común donde toda persona que busca datos cae, es el internet. El internet se comporta como un océano de datos, océano con corrientes en muchas direcciones, tantas direcciones como ciencias que el investigador desee profundizar. Está en sus objetivos poder identificar las corrientes que le permita hacerse de un acervo de datos que permita cumplir los objetivos de su investigación.

Antes de avanzar, algunos conceptos son importantes. La datificación es inevitable, y definir este gran cúmulo de zettabytes es necesario. “Big Data o datos a gran escala hace referencia a un conjunto de datos tan grande que las aplicaciones informáticas tradicionales de procesamiento de datos no son capaces de tratar con ellos ni de encontrar patrones repetitivos” [1].

*Autor en correspondencia.

La asistencia de un lenguaje de programación para estos propósitos es necesario. “Python es un lenguaje de programación de alto nivel que se caracteriza por el hecho de ser un lenguaje simple, fácil de leer, escribir y depurar, y además es portable. Sin embargo, una característica básica es la de ser un lenguaje interpretado” [2]. Este ensayo se basará en Python como lenguaje de adquisición de datos, y en una librería específica para el fin propuesto, BeautifulSoup.

2 Objetivos

2.1 Objetivo General

- Diseñar una propuesta metodológica para el diseño de algoritmos de web scraping con Beautiful Soup.

2.2 Objetivos Específicos

- Establecer criterios para definir la posibilidad de que una web sea candidata a web scraping.
- Elaborar un algoritmo que permita la construcción de una base de datos tabular con formato csv asistido por pandas.
- Aplicar la metodología diseñada por medio de un ejemplo aplicado.

3 Metodología propuesta

Como propuesta metodológica para este trabajo, se utilizará la “V” de la Big Data [1].

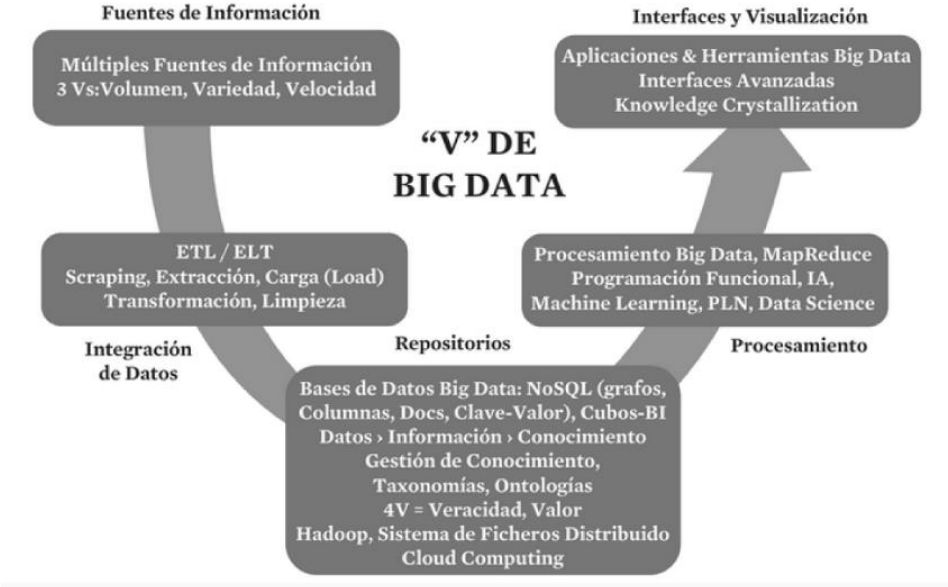

Fig. 1. Mapa de Proceso en Big Data.
Fuente: Ortega Candel, J. M. (2022) [1].

El *Scraping* (o *web scraping*) es una técnica de la minería de datos y de la ciencia de datos que permite, por medio de algoritmos y librerías específicas, la adquisición de datos publicados en la web [3]. Estos datos, luego pueden ser sistematizados para poder aplicar nuevas técnicas con estos datos. Esta técnica permite obtener el contenido de las páginas web, y extraer información útil de ellas [4]. Otra de las características del *scraping*, es que puede ser una tarea programada, para así recuperar información de la web de manera automática [5]. Para la aplicación de esta herramienta, es importante adquirir múltiples fuentes de información, para poder seleccionar las webs que se utilizarán para *scrapear*.

Las fuentes de información deben cumplir algunas características, según la experiencia de estos autores:

- El sitio debe ser visible sin necesidad de iniciar alguna sesión. En caso de necesitar conexión, es probable que un token o el consumo por medio de una API puede hacer el trabajo más fácil.
- La información debe poder ser adquirida en texto. Las imágenes suelen generar complicaciones, y se requeriría una técnica adicional para poder extraer el texto, como una CNN (Convolutional Neuronal Network).
- En caso de publicación de los datos y por cuestiones legales y de privacidad, se debe asegurar que los datos son públicos.

Seleccionada la web, se procede a la manipulación y adquisición de los datos. Las páginas web hoy día utilizan estructuras de datos, de donde es importante que se ubique de dónde se extraerán. Para ello, es importante inspeccionar la web que se está manipulando. Identificados los datos, es posible hacer un proceso de transformación de los datos, para poder limpiarlo y guardarlo con el fin que tenga la investigación donde se aplicará.

Almacenar los datos es un paso importante. Es habitual que los datos se adquieran y exporten en un formato CSV (archivo separado por comas). Estos archivos luego pueden pasar a ser parte de bases de datos relacionales o no relacionales, para poder gestionarlas a partir de sistemas de gestión de bases de datos. Adquirir un CSV con datos limpios es un buen inicio.

Los pasos de Procesamiento, Interfaces y Visualización se omitirán en este trabajo, ya que quedará en responsabilidad de cada investigador o científico de datos su implementación.

4 Resultados y Discusiones

4.1 Fuentes de Información

La temática de elección para el diseño del algoritmo de web scraping, ha sido el clima sinóptico de la ciudad de Posadas. Las fuentes de información fueron variadas, por lo que se procedió a evaluar los criterios propuestos anteriormente:

- Datos Públicos
- Información en texto
- Descarga anónima

Las fuentes de datos candidatas fueron:

- **Accuweather**. URL: https://www.accuweather.com/es/.
- **Wunder Underground**. URL: https://www.wunderground.com/.
- **Yahoo Clima**. URL: https://www.yahoo.com/news/weather.
- **YoWindow**. URL: https://yowindow.com/.
- **WeatherBug**. URL: https://www.weatherbug.com/.

Tabla 1. Comparativa entre sitios web para la ejecución del scraping.
Fuente: Elaboración propia.

	Público	Texto	Anónimo	Comentarios
Accuweather	Sí	La mayoría	Sí	Tiene imágenes y texto combinados, complejo para extraer.
Wunder Underground	Sí	Sí	Sí	La información está tabulada, lista para extraer.
Yahoo Clima	Sí	La mayoría	Sí	El diseño interactivo de la aplicación web puede presentar dificultades para el algoritmo.
YoWindow	No	Una parte	No	Se debe accede por medio de una aplicación web y requiere un logueo.
WeatherBug	Sí	La mayoría	Sí	El diseño interactivo de la aplicación web puede presentar dificultades para el algoritmo.

Se seleccionó Wunder Underground como el sitio para ejecutar el scraping. Para ello, se seleccionó una estación meteorológica, se analizó el link, y se inspeccionó la web [6, 7]. El formato del link es:

https://www.wunderground.com/dashboard/pws/**IPOSAD6**/table/**2024-05-27**/**2024-05-27**/daily

donde **IPOSAD6 es la estación meteorológica que aporta los datos** y **2024-05-27 es el día de la consulta en formato aaaa-mm-dd**.

Para la elaboración del algoritmo, se tomarón dos estaciones meteorológicas, IPOSAD6 e IPOSAD12, en el período de 2 años (2022 y 2023).

4.2 Integración de Datos

Una vez seleccionada la web a trabajar el scrapping, se procede a hacer la construcción del algoritmo. Se utilizó Python por ser un lenguaje generalista de alto nivel. Para ello, la sintaxis general fue:

- Importación de librerías: csv, requests, BeautifulSoup (de bs4), pandas, numpy, datatime, os, openpyxl, pydotplus.
- Definición de la cantidad de días que tienen los meses en cuestión para fijar los bucles.
- Construcción de un dataframe donde se guardarán los datos.
- Realización de bucles donde los parámetros que van cambiando entre iteraciones, son la fecha y la estación. Ambos cambios se realizan en el link analizado previamente.
- Realizado el recorrido, se procede a realizar varios requests sobre los links (con los cambios explicados anteriormente) y se analiza la web con Beatutiful Soup.
- Programación de un try-except, para el manejo de errores, por si el algoritmo tiene problemas de reconocimiento de los datos y su posterior almacenado.
- Finalmente, se procede a limpiar los datos, dejando los números en específico, y eliminando información superflua, como la unidad de medida de la variable analizada.

Una vez limpios los datos, se procede a la construcción de tablas.

4.3 Repositorios

La librería Pandas es una excelente aliada para este trabajo. Para ello, se utilizó Pandas para crear dataframes. Todos los datos limpios con la fracción de código anterior, se procede a guardar en un dataframe de Pandas. Una vez concatenadas las tablas, se exporta los datos a un archivo CSV, que funcionará como insumo para los posteriores análisis que se requieran hacer [8, 9].

En caso se seguir con los pasos 4 y 5 de la metodología, se pueden usar librerías como matplotlib, seaborn, plotly, scikit Learn, entre las más conocidas. Con ellas, será posible sacar conclusiones de los datos adquiridos.

Poder realizar todo el trabajo anteriormente descripto, ha llevado a lograr los siguientes resultados:

Fig. 2. Infografía resumen de la experiencia corriendo el algoritmo de web scraping.
Fuente: Elaboración propia.

Los datos recopilados fueron almacenados como strings, que además incluían la unidad de medida. Para poder trabajar los datos, fue necesario limpiar los mismos.
Las columnas obtenidas han sido:

- temperatura: Temperatura de bulbo seco medida en la estación meteorológica. Medida en °F.
- puntoderocio: Temperatura de bulbo húmedo medida en la estación meteorológica. Medida en °F.
- humedad: Humedad del aire. Medida en porcentaje.
- viento: Dirección del Viento, cuyo valor es un punto cardinal. Variable cualitativa.
- velocidad: Velocidad del Viento. Medida en mph.
- rafaga: Ráfaga de Viento. Medida en mph.
- presion: Presión atmosférica medida en la estación meteorológica. Medida en in.
- preciprate: Ratio de precipitaciones. Medida en in.
- precipacum: Precipitaciones acumuladas. Medida en in.
- uv solar: Intensidad de rayos ultravioleta captados en la estación meteorológica. Medida en escala UV.
- iposad: Estación de donde se extraen los datos. Puede ser 6 u 12.
- fecha_hora: Fecha y hora de la toma de la muestra.

No se han presentado datos perdidos o nulos.

Tabla 2. Cabecera de la tabla de los datos y sus variables.
Fuente: Elaboración Propia.

	temperatura	puntoderocio	humedad	viento	velocidad	rafaga	presion	preciprate	precipacum	uv	solar	fecha_hora	iposad
0	84.4	72.6	68	ENE	0.4	0.7	29.89	0.0	0.0	0	0.0	2022-01-01 00:04:00	6
1	84.2	71.9	67	North	0.6	0.8	29.88	0.0	0.0	0	0.0	2022-01-01 00:09:00	6
2	84.2	71.8	67	NNE	0.2	0.4	29.89	0.0	0.0	0	0.0	2022-01-01 00:14:00	6
3	83.7	71.6	67	NNE	0.0	0.1	29.89	0.0	0.0	0	0.0	2022-01-01 00:19:00	6
4	83.6	71.0	66	NNE	0.2	0.5	29.89	0.0	0.0	0	0.0	2022-01-01 00:24:00	6

En los Anexos, se encuentran los códigos fuente utilizados para este fin.

Así, y a modo de síntesis de todo el trabajo realizado, podemos resumir los pasos en la siguiente propuesta metodológica:

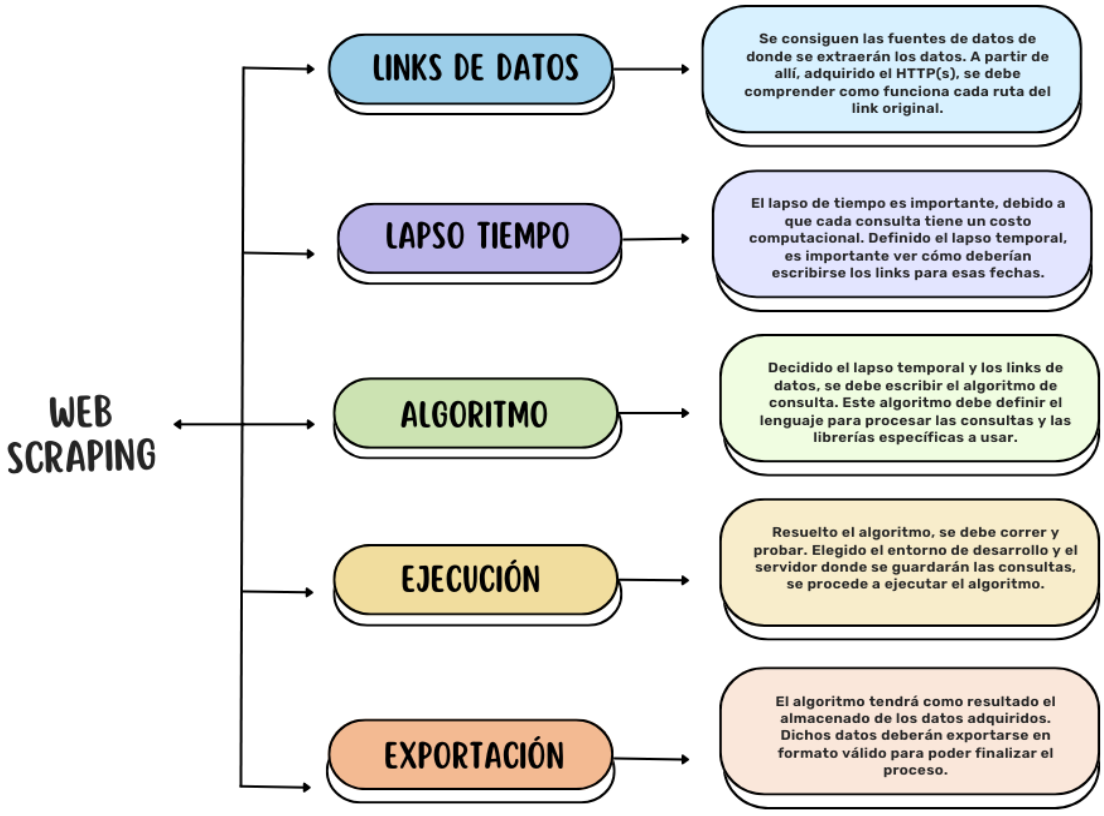

Fig. 3. Infografía resumen de la propuesta metodológica para la elaboración de algoritmos de web scraping. **Fuente:** Elaboración propia.

5 Conclusiones

A partir de este trabajo podemos concluir lo siguiente:

- Es posible establecer criterios para la selección de webs candidatas para web scraping.
- La adquisición de los datos por medio de web scrapping es efectiva, sin embargo requiere un esfuerzo computacional considerable y un posterior trabajo de limpieza de los datos.
- Sistematizar los pasos para la obtención de datos por scraping es posible.
- Se logró diseñar una propuesta metodológica como una primera aproximación a la minería de datos a partir de la aplicación de la técnica de web scraping.

Como sugerencias para futuros trabajos, es interesante pensar en buscar nuevos criterios de clasificación de webs candidatas, y comparar los modelos de obtención de datos, ya sea con algoritmos distintos, o con diferentes librerías.

Referencias

[1] Ortega Candel, J. M. (2022). Big data, machine learning y data science en Python: (1 ed.). RA-MA Editorial. https://elibro.net/es/ereader/elibrounam/230290?page=8.

[2] Sarasa Cabezuelo, A. (2017). Gestión de la información web usando Python: (ed.). Barcelona, Editorial UOC. Recuperado de https://elibro.net/es/ereader/elibrounam/114201?page=10 el 05 de Junio de 2024.

[3] Gábor László Hajba (2018). "Website Scraping with Python: Using BeautifulSoup and Scrapy". Apress Berkeley, eBook ISBN 978-1-4842-3925-4CA. https://doi.org/10.1007/978-1-4842-3925-4.

[4] Ryan Mitchel (2024). " Web Scraping with Python, 3rd Edition". O'Reilly Media, Inc. ISBN: 9781098145354

[5] Jacqueline Kazil, Katharine Jarmul (2016). "Data Wrangling with Python". O'Reilly Media, Inc. ISBN: 9781491948774

[6] Base de Datos OPAD del Centro de Posadas: https://www.wunderground.com/dashboard/pws/IPOSAD6. (consultado entre el 01 al 15 de Abril de 2024)

[7] Base de Datos OPAD de Itaembé Miní: https://www.wunderground.com/dashboard/pws/IPOSAD12. (consultado entre el 01 al 15 de Abril de 2024)

[8] Wes McKinney (2017). "Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython". Editor O'Reilly Media, Inc. ISBN 9781491957639

[9] Trevor Hastie, Robert Tibshirani, Jerome Friedman (2009). “The Elements of Statistical Learning, Data Mining, Inference, and Prediction, Second Edition”. Springer New York, NY. https://doi.org/10.1007/978-0-387-84858-7.

Anexos

ANEXO 1: Algoritmo de Web Scraping en Python

```
import csv
import requests
from bs4 import BeautifulSoup
import pandas as pd

dia_maximo={
        1:31,
        2:28,
        3:31,
        4:30,
        5:31,
        6:30,
        7:31,
        8:31,
        9:30,
        10:31,
        11:30,
        12:31
}

df_clima = None

for anio in range(2022, 2024):
   for mes in range(12):
      dias = dia_maximo[mes+1]
      for dia in range(dias):
         if mes < 9:
            mes_ = '0' + str(mes+1)
         else:
            mes_ = str(mes+1)
         if dia < 9:
            dia_ = '0' + str(dia+1)
         else:
```

```
            dia_ = str(dia+1)
        url = f'https://www.wunderground.com/dashboard/pws/IPOSAD12/table/{anio}-{mes_}-{dia_}/{anio}-{mes_}-{dia_}/daily'
        print(url)
        try:
            req = requests.get(url)
            soup = BeautifulSoup(req.text, 'html.parser')
            data = soup.find_all("table")[3]
            df = pd.read_html(str(data))[0]
            # Agrega una columna adicional con la fecha de la consulta actual
            df['Fecha'] = f'{anio}-{mes_}-{dia_}'
            if df_clima is None:
                df_clima = df
            else:
                df_clima = pd.concat([df_clima, df], axis=0)
        except Exception as e:
            print(f"Error procesando URL {url}: {e}. Continuando con la siguiente...")
            continue

# Reinicializa el índice del DataFrame después de procesar todos los días del año
if df_clima is not None:
    df_clima = df_clima.reset_index(drop=True)
    print(df_clima)
    # Exporta el DataFrame a un archivo Excel
    df_clima.to_excel("clima.xlsx", index=False)
    print("Datos exportados a clima.xlsx.")
else:
    print("No se recopilaron datos.")

Algoritmo de Limpieza de los Datos
import pandas as pd
import seaborn as sns
import datetime

df6 = pd.read_excel('IPOSAD6 2022 2023.xlsx', engine='openpyxl')
df12 = pd.read_excel('IPOSAD12 2022 2023.xlsx', engine='openpyxl')

df6.head()

df12.head()

print(df6.shape)
print(df12.shape)

valores_faltantes = df6.isna().sum()
print(valores_faltantes)

df6=df6.dropna()
valores_faltantes = df6.isna().sum()
print(valores_faltantes)

valores_faltantes12 = df12.isna().sum()
print(valores_faltantes12)

df12=df12.dropna()
```

```
valores_faltantes12 = df12.isna().sum()
print(valores_faltantes12)

nuevos_nombres_cokumnas = ['hora', 'temperatura', 'puntoderocio','humedad', 'viento', 'velocidad', 'rafaga', 'presion',
'preciprate','precipacum', 'uv', 'solar', 'fecha', 'iposad']
df6.columns = nuevos_nombres_cokumnas
df12.columns = nuevos_nombres_cokumnas

df_unido = pd.concat([df6, df12], ignore_index=True)

df_unido.head()

#Distribución de clases
distribucion_clases = df_unido["iposad"].value_counts()
print(distribucion_clases)

sns.catplot(data=df_unido, x='iposad', kind='count');

df_unido['hora'] = pd.to_datetime(df_unido['hora'], format='%I:%M %p')
df_unido['hora'] = df_unido['hora'].dt.strftime('%H:%M')

df_unido.head()

df_unido['fecha'] = pd.to_datetime(df_unido['fecha'])
df_unido['fecha_hora'] = df_unido.apply(lambda row: datetime.datetime.combine(row['fecha'], row['hora']), axis=1)
df_unido.drop(['fecha', 'hora'], axis=1, inplace=True)
print(df_unido.head())

df_unido.to_excel('data_unida.xlsx', index=False)
```

ANEXO 2: Preprocesamiento de los datos

```
#1. Importación de Librerías
"""

import pandas as pd
import numpy as np
import os
import random
!pip install graphviz pydotplus
!pip install openpyxl
import openpyxl
import graphviz
import pydotplus


"""# 2. Carga de Archivo"""

os.makedirs('media', exist_ok=True)
file_id = '18s2k2OkonYnAHZO6vDhPs991f9epc0TC'
destination = 'data_unida.xlsx'
!gdown --id $file_id -O $destination

df = pd.read_excel('data_unida.xlsx', engine='openpyxl')
df.head()
```

```
# Se reorganizan las columnas
column_order = ['temperatura', 'puntoderocio', 'humedad', 'viento', 'velocidad',
                'rafaga', 'presion', 'preciprate', 'precipacum', 'uv', 'solar',
                'fecha_hora', 'iposad']
df = df.reindex(columns=column_order)
df

print(df.dtypes)

df.replace("--", np.nan, inplace=True)
columnas_a_convertir = ['temperatura', 'puntoderocio', 'velocidad', 'rafaga', 'presion', 'preciprate', 'precipacum', 'solar']

# Se convierten las columnas especificadas a tipo float
df[columnas_a_convertir] = df[columnas_a_convertir].astype(float)
df = df.dropna()
df["humedad"] = df["humedad"].astype(int)

# Se verifican los tipos de datos después de la conversión
print(df.dtypes)

df
```