

Autor: María José Gómez Silva

# Machine Learning con Python. Semana 1.

## CAPÍTULO 2. INGESTA DE DATOS.

### 2. 2 Estructuras e Ingesta de Datos en Pandas

## ESTRUCTURAS DE DATOS EN PANDAS, SERIES Y DATAFRAMES

En pandas disponemos de varias estructuras de datos y operaciones (métodos) para manipular dichas estructuras de forma sencilla y eficiente. Las dos estructuras de datos principales son las *Series* (datos en una dimensión) y los *DataFrames* (datos en dos dimensiones), capaces de representar secuencias de datos y datos en forma tabular, respectivamente.

#### Series

Una variable del tipo *Series* es una variable de una dimensión que contiene datos de un cierto tipo y tiene asociado un índice o etiqueta para cada dato.

La función *pandas.Series* es la constructora de series (la documentación está disponible en: <a href="https://pandas.pydata.org/docs/reference/api/pandas.Series.html">https://pandas.pydata.org/docs/reference/api/pandas.Series.html</a>).

Como se puede observar en la Figura 1, la serie s contiene elementos de tipo int64. Los elementos de la serie están indexados mediante un array de enteros comenzando desde el cero (columna de la izquierda).

Figura 1. Creación de una estructura de tipo Series

Además, las Series permiten indexar cada uno de sus elementos con un valor descriptivo o etiqueta. Por ejemplo, si nuestros datos son el número de días que tiene cada mes, será de gran utilidad usar el nombre de los meses como etiquetas del índice, en lugar de etiquetas de tipo entero. Con el argumento *name* podemos asignar un nombre a la serie, que es posible consultar a posteriori. También, podemos acceder a los valores y los índices de la serie por separado, con los métodos *values* e *index*, como se observa en el ejemplo de la Figura 2.

```
Out[2]: Ene
     Mar
         31
     Abr
         30
     May
         31
     Jun
         30
     Jul
         31
     Ago
     Sep
         30
     0ct
         31
     Nov
         30
     Dic
         31
     Name: Días de cada mes, dtype: int64
In [3]: dias.name
Out[3]: 'Días de cada mes'
In [7]: dias.values
Out[7]: array([31, 28, 31, 30, 31, 30, 31, 30, 31, 30, 31], dtype=int64)
In [8]: dias.index
dtype='object')
```

Figura 2. Creación de una serie con etiquetas para los índices y nombre, y uso de los métodos name, values e index.

#### **DataFrame**

Las estructuras de tipo *DataFrame* están diseñadas para manejar datos representados en forma de tabla, donde tanto las filas como las columnas están indexadas. Cada columna puede tener asociado un tipo de dato diferente. Un *DataFrame* puede entenderse como una colección de *Series*, ya que cada columna es una estuctrura de tipo *Series*.

Con la función *pandas.DataFrame* es posible crear un *DataFrame* a partir de los datos contenidos en una lista, un diccionario, una *serie* o en otro *DataFrame*. En la Figura 3 se crea un *DataFrame* a partir de un diccionario de python. De forma similar a como se hacía en las *Series*, es posible acceder a los valores del *DataFrame*. Sin embargo, ahora no se tiene un único índice asociado a cada fila (*index*), sino que también se tiene un índice para las columnas (*columns*).

```
tabla
 Out[9]:
              Mes Días Estación Temp.2021 Temp.2022
           Enero 31
       Ene
            Febrero
       Feb
       Mar
       Abr
                  30
                       pri
       May
             Mayo 31 pri
             Junio
                 30
                       pri
            Agosto 31
                       ver
       Ago
                           25 18
       Nov Noviembre 30 oto 11 10
In [10]: tabla.values
In [11]: tabla.index
Out[11]: Index(['Ene', 'Feb', 'Mar', 'Abr', 'May', 'Jun', 'Jul', 'Ago', 'Sep', 'Oct', 'Nov', 'Dic'], dtype='object')
In [12]: tabla.columns
Out[12]: Index(['Mes', 'Días', 'Estación', 'Temp.2021', 'Temp.2022'], dtype='object')
```

Figura 3. Creación de un DataFrame a partir de un diccionario y uso de los métodos values, index y columns

#### INGESTA DE DATOS CON PANDAS

Pandas proporciona funciones para realizar la lectura de una gran variedad de fuentes de datos y almacenar su contenido en estructuras *DataFrame*.

## Lectura y escritura de ficheros HTML

Los métodos *pd.read\_html* y *pd.to\_html* de Pandas se emplean, respectivamente, para leer y escribir ficheros en formato HTML.

El método *pd.read\_html* recorre un fichero HTML en busca de tablas, y devuelve una lista de *DataFrames*, uno por cada tabla encontrada. Previamente, para poder acceder al código HTML de una página web hay que realizar una petición con protocolo HTTP Request/Response, como se muestra en la Figura 4.



Figura 4. Lectura de datos en formato HTML

## Lectura y escritura de archivos JSON

Los métodos *pd.read\_json* y *pd.to\_json* de Pandas se emplean, respectivamente, para leer y escribir ficheros en formato JSON. En la Figura 5 se muestra como una *DataFrame*, previamente creado, es guardado en un archivo JSON, cuyo contenido se muestra en la Figura 6. Seguidamente, la Figura 7 muestra la operación inversa, de lectura de un archivo JSON.

```
tabla
Out[88]:
           Mes Días Estación Temp.2021 Temp.2022
      Ene
           Enero
      Feb
           Marzo
                               10
                          15
      Jul
                      20
           Julio 31
                               24
                               26
              30
                      25
                               18
      Oct
          Octubre
               31
                   oto
                         20
                               14
                        11
                  oto
      Nov Noviembre 30
                               10
      Dic
         Diciembre 31
                   oto
                          6
In [89]: import json
     tabla.to_json('meses.json')
```

Figura 5. Escritura de un archivo JSON a partir de un DataFrame

```
{"Mes":{
          "Ene":"Enero","Feb":"Febrero","Mar":"Marzo","Abr":"Abril",
"May":"Mayo","Jun":"Junio","Jul":"Julio","Ago":"Agosto",
"Sep":"Septiembre","Oct":"Octubre","Nov":"Noviembre","Dic":"Diciembre"
          },
 "D\u00edas":{
          "Ene":31,"Feb":28,"Mar":31,"Abr":30,"May":31,"Jun":30,
          "Jul":31, "Ago":31, "Sep":30, "Oct":31, "Nov":30, "Dic":31
 "Estaci\u00f3n":{
          "Ene":"inv","Feb":"inv","Mar":"inv","Abr":"pri",
"May":"pri","Jun":"pri","Jul":"ver","Ago":"ver",
          "Sep":"ver","Oct":"oto","Nov":"oto","Dic":"oto"
          },
 "Temp.2021":{
          "Ene":2, "Feb":3, "Mar":7, "Abr":7, "May":9, "Jun":15,
          "Jul":20,"Ago":27,"Sep":25,"Oct":20,"Nov":11,"Dic":6
          },
 "Temp.2022":{
          "Ene":4, "Feb":2, "Mar":5, "Abr":9, "May":10, "Jun":14,
          "Jul":24, "Ago":26, "Sep":18, "Oct":14, "Nov":10, "Dic":7
```

Figura 6. Archivo JSON generado.

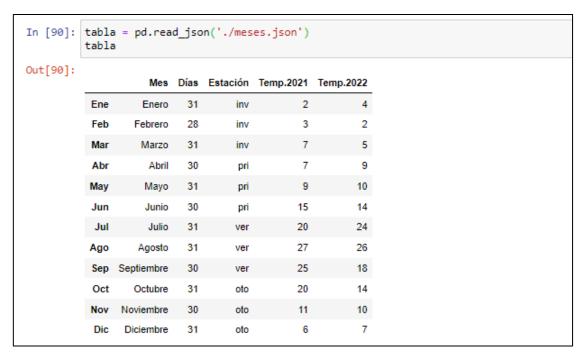


Figura 7. Lectura de un archivo JSON como DataFrame

## Lectura y escritura de ficheros CSV y XLSX

Con las funciones *pd.read\_csv* y *pd.read\_excel* es posible generar *DataFrames* a partir de los datos contenidos en archivos de tipo CSV y XLSX (y xls), respectivamente. A su vez, un *DataFrame* puede ser guardado en formato CSV y XLSX con los métodos *pd.to\_csv* y *pd.to\_excel*, respectivamente.

Las funciones *read\_csv* y *read\_excel* son flexibles y permiten saltar cabeceras, saltar filas, leer un número determinado de filas o columnas, dar nuevos nombres a las columnas, etc. Su funcionamiento depende de los valores dados a los argumentos (*header, names, skiprows, index\_col, nrows*) de las funciones.

En la Figura 8 se muestra la generación de un archivo CSV a partir de un *DataFrame* y posteriormente se vuelve a leer el CSV como *DataFrame*, empleando diferentes opciones como índices. En la última ejecución del ejemplo se usan varias columnas como índices jerárquicos.

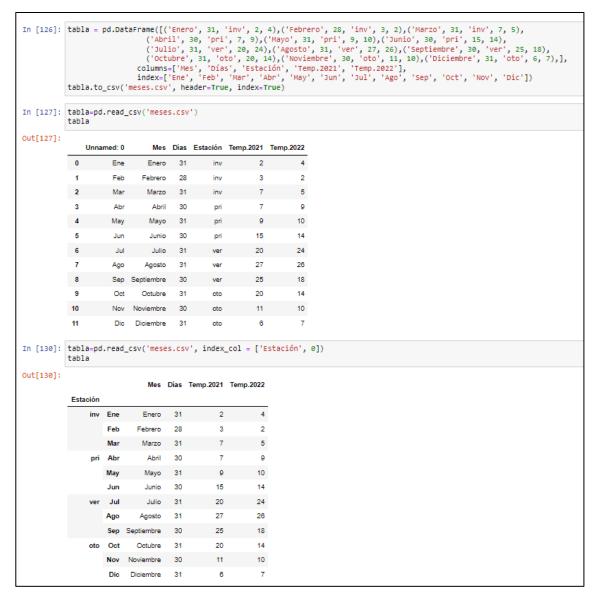


Figura 8. Escritura y lectura de un archivo CSV.

En la Figura 9 se muestra un ejemplo en el que se crea un *DataFrame* a partir del archivo *Superstore\_Dataset.xlsx*, disponible en el campus virtual. Este archivo contiene datos de pedidos y devoluciones de unos grandes almacenes en dos hojas de cálculo, *Orders* y *Returns*, respectivamente. En el ejemplo de la Figura 9 se cargan únicamente los datos de la hoja *Orders*. Por defecto, si no se indica nada, la función *read\_excel* lee los datos de la primera hoja dentro del fichero.

Las bases de datos contenidas en archivos suelen tener una gran extensión. El método *head* visualiza únicamente las cinco primeras filas de la tabla.

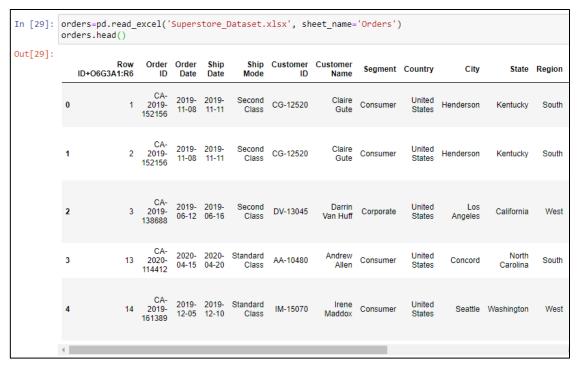


Figura 9. Cinco primeras filas del DataFrame generado a partir de un archivo xlsx.

Si se necesita trabajar con varias hojas de cálculo de Excel, la clase *ExcelFile* de Pandas es más eficiente, ya que carga el fichero completo en memoria una única vez y permite crear posteriormente *DataFrames* a partir de cada hoja del libro, como se muestra en la Figura 10.

```
In [54]: libro = pd.ExcelFile('Superstore_Dataset.xlsx')
    orders = pd.read_excel(libro, sheet_name = 'Orders')
    returns = pd.read_excel(libro, sheet_name = 'Returns')
```

Figura 10. Creación de dos DataFrames a partir de distintas hojas de cálculo de un libro Excel.

## **REFERENCIAS**

https://docs.python.org/3/tutorial/index.html

https://www.anaconda.com/products/individual

https://docs.jupyter.org/en/latest/

https://numpy.org/doc/stable/

https://pandas.pydata.org/docs

https://matplotlib.org/stable/index.html

Machine learning con Python y Scikit-learn by Joaquín Amat Rodrigo, available under a Attribution 4.0 International (CC BY 4.0) at https://www.cienciadedatos.net/documentos/py06\_machine\_learning\_python\_scikitlearn.htm

https://www.drivendata.org/