Autor: María José
Gómez Silva

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: https://pandas.pydata.org/docs/reference/api/pandas.Series.html).

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).

```
In [1]: import pandas as pd

        s = pd.Series([0,1,4,9,16,25])
        s

Out[1]: 0     0
        1     1
        2     4
        3     9
        4    16
        5    25
        dtype: int64
```

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.

```
In [2]: dias = pd.Series([31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31],
                         index=['Ene', 'Feb', 'Mar', 'Abr', 'May', 'Jun', 'Jul', 'Ago', 'Sep', 'Oct', 'Nov', 'Dic'],
                         name='Días de cada mes')
        dias

Out[2]: Ene    31
        Feb    28
        Mar    31
        Abr    30
        May    31
        Jun    30
        Jul    31
        Ago    31
        Sep    30
        Oct    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, 31, 30, 31, 30, 31], dtype=int64)

In [8]: dias.index

Out[8]: Index(['Ene', 'Feb', 'Mar', 'Abr', 'May', 'Jun', 'Jul', 'Ago', 'Sep', 'Oct',
               'Nov', 'Dic'],
              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*).

```
In [9]: tabla = pd.DataFrame([('Enero', 31, 'inv', 2, 4),('Febrero', 28, 'inv', 3, 2),('Marzo', 31, 'inv', 7, 5),
                              ('Abril', 30, 'pri', 7, 9),('Mayo', 31, 'pri', 9, 10),('Junio', 30, 'pri', 15, 14),
                              ('Julio', 31, 'ver', 20, 24),('Agosto', 31, 'ver', 27, 26),('Septiembre', 30, 'ver', 25, 18),
                              ('Octubre', 31, 'oto', 20, 14),('Noviembre', 30, 'oto', 11, 10),('Diciembre', 31, 'oto', 6, 7),],
                             columns=['Mes', 'Días', 'Estación', 'Temp.2021', 'Temp.2022'],
                             index=['Ene', 'Feb', 'Mar', 'Abr', 'May', 'Jun', 'Jul', 'Ago', 'Sep', 'Oct', 'Nov', 'Dic'])
        tabla
```

Out[9]:

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

```
In [10]: tabla.values
```

```
Out[10]: array([['Enero', 31, 'inv', 2, 4],
                ['Febrero', 28, 'inv', 3, 2],
                ['Marzo', 31, 'inv', 7, 5],
                ['Abril', 30, 'pri', 7, 9],
                ['Mayo', 31, 'pri', 9, 10],
                ['Junio', 30, 'pri', 15, 14],
                ['Julio', 31, 'ver', 20, 24],
                ['Agosto', 31, 'ver', 27, 26],
                ['Septiembre', 30, 'ver', 25, 18],
                ['Octubre', 31, 'oto', 20, 14],
                ['Noviembre', 30, 'oto', 11, 10],
                ['Diciembre', 31, 'oto', 6, 7]], dtype=object)
```

```
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.

```
In [58]: import requests
         import pandas as pd
         url = "https://es.wikipedia.org/wiki/Europa"
         respuesta = requests.get(url)
         if respuesta.status_code == 200:
             print('Ok')
         else:
             print( 'No ok')

         Ok

In [59]: codigoHTML = respuesta.text
         lista_df = pd.read_html(codigoHTML, header=0)
         len(lista_df)

Out[59]: 10

In [61]: lista_df[1].head(4)

Out[61]:
```

	Bandera	Nombre/Nombre oficial	Establecido	Superficie(km²)	Población	Habitantes por km²	Capital
0	NaN	Albania República de Albania	1912	28 748	3 038 594	1056	Tirana
1	NaN	Alemania República Federal de Alemania	1871	357 022	80 722 792	2261	Berlín
2	NaN	Andorra Co-Principado de Andorra	1278	468	85 660	183	Andorrala Vieja
3	NaN	Armenia República de Armenia	1991	29 749	3 229 900	1085	Ereván

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.

```
In [88]: tabla = pd.DataFrame([('Enero', 31, 'inv', 2, 4),('Febrero', 28, 'inv', 3, 2),('Marzo', 31, 'inv', 7, 5),
                               ('Abril', 30, 'pri', 7, 9),('Mayo', 31, 'pri', 9, 10),('Junio', 30, 'pri', 15, 14),
                               ('Julio', 31, 'ver', 20, 24),('Agosto', 31, 'ver', 27, 26),('Septiembre', 30, 'ver', 25, 18),
                               ('Octubre', 31, 'oto', 20, 14),('Noviembre', 30, 'oto', 11, 10),('Diciembre', 31, 'oto', 6, 7),],
                              columns=['Mes', 'Días', 'Estación', 'Temp.2021', 'Temp.2022'],
                              index=['Ene', 'Feb', 'Mar', 'Abr', 'May', 'Jun', 'Jul', 'Ago', 'Sep', 'Oct', 'Nov', 'Dic'])
         tabla
```

Out[88]:

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

```
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.

```
In [90]: tabla = pd.read_json('./meses.json')
         tabla
```

Out[90]:

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

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.

```
In [126]: tabla = pd.DataFrame([('Enero', 31, 'inv', 2, 4),('Febrero', 28, 'inv', 3, 2),('Marzo', 31, 'inv', 7, 5),
                                ('Abril', 30, 'pri', 7, 9),('Mayo', 31, 'pri', 9, 10),('Junio', 30, 'pri', 15, 14),
                                ('Julio', 31, 'ver', 20, 24),('Agosto', 31, 'ver', 27, 26),('Septiembre', 30, 'ver', 25, 18),
                                ('Octubre', 31, 'oto', 20, 14),('Noviembre', 30, 'oto', 11, 10),('Diciembre', 31, 'oto', 6, 7),],
                               columns=['Mes', 'Días', 'Estación', 'Temp.2021', 'Temp.2022'],
                               index=['Ene', 'Feb', 'Mar', 'Abr', 'May', 'Jun', 'Jul', 'Ago', 'Sep', 'Oct', 'Nov', 'Dic'])
          tabla.to_csv('meses.csv', header=True, index=True)

In [127]: tabla=pd.read_csv('meses.csv')
          tabla

Out[127]:
```

	Unnamed: 0	Mes	Días	Estación	Temp.2021	Temp.2022
0	Ene	Enero	31	inv	2	4
1	Feb	Febrero	28	inv	3	2
2	Mar	Marzo	31	inv	7	5
3	Abr	Abril	30	pri	7	9
4	May	Mayo	31	pri	9	10
5	Jun	Junio	30	pri	15	14
6	Jul	Julio	31	ver	20	24
7	Ago	Agosto	31	ver	27	26
8	Sep	Septiembre	30	ver	25	18
9	Oct	Octubre	31	oto	20	14
10	Nov	Noviembre	30	oto	11	10
11	Dic	Diciembre	31	oto	6	7

```
In [130]: tabla=pd.read_csv('meses.csv', index_col = ['Estación', 0])
          tabla

Out[130]:
```

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

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.

```
In [29]: orders=pd.read_excel('Superstore_Dataset.xlsx', sheet_name='Orders')
         orders.head()
```

Out[29]:

	Row ID+O6G3A1:R6	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	State	Region
0	1	CA-2019-152156	2019-11-08	2019-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	Kentucky	South
1	2	CA-2019-152156	2019-11-08	2019-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	Kentucky	South
2	3	CA-2019-138688	2019-06-12	2019-06-16	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	California	West
3	13	CA-2020-114412	2020-04-15	2020-04-20	Standard Class	AA-10480	Andrew Allen	Consumer	United States	Concord	North Carolina	South
4	14	CA-2019-161389	2019-12-05	2019-12-10	Standard Class	IM-15070	Irene Maddox	Consumer	United States	Seattle	Washington	West

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.html

https://www.drivendata.org/