

UNIVERSIDAD  
COMPLUTENSE  
DE MADRID



Autor: María José  
Gómez Silva

# Machine Learning con Python. Semana 1.

## CAPÍTULO 2. INGESTA DE DATOS.

### 2. 3 Manejo de los Datos en Pandas

A continuación, se presentan algunos mecanismos para acceder y actualizar los datos contenidos en estructuras de tipo Series y de tipo DataFrame, y se introducen algunas de las operaciones que es posible realizar sobre esos datos, como las de ordenación y las aritméticas.

#### ACCESO

El uso de etiquetas para indexar los elementos, tanto de una Serie como de un DataFrame, permite acceder a los datos con dos procedimientos: usando notación de corchetes, o usando la etiqueta como si fuera una propiedad de la serie. Además, en este caso, sería posible consultar si una determinada etiqueta está entre los índices, como en el ejemplo de la Figura 1. La Figura 1 muestra el acceso a elementos de una Serie, y la Figura 2 muestra el acceso a columnas de un DataFrame.

```
In [3]: 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['Abr']

Out[3]: 30

In [4]: dias.Abr

Out[4]: 30

In [5]: 'Nov' in dias.index

Out[5]: True
```

*Figura 1. Acceso a un elemento de la serie y comprobación de la presencia de una determinada etiqueta.*

```
In [13]: 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.Mes

Out[13]: Ene      Enero
Feb      Febrero
Mar      Marzo
Abr      Abril
May      Mayo
Jun      Junio
Jul      Julio
Ago      Agosto
Sep      Septiembre
Oct      Octubre
Nov      Noviembre
Dic      Diciembre
Name: Mes, dtype: object

In [14]: tabla['Mes']

Out[14]: Ene      Enero
Feb      Febrero
Mar      Marzo
Abr      Abril
May      Mayo
Jun      Junio
Jul      Julio
Ago      Agosto
Sep      Septiembre
Oct      Octubre
Nov      Noviembre
Dic      Diciembre
Name: Mes, dtype: object
```

Figura 2. Acceso a una columna de un DataFrame.

Para acceder a partes de un DataFrame se emplean los indexadores *loc* e *iloc*. El método *loc* permite una selección por etiquetas, es decir, permite acceder a partes de un DataFrame indicando el nombre de las etiquetas correspondientes a las filas y columnas a las que queremos acceder. El método *iloc* permite la selección por posición, es decir, permite acceder a partes de un DataFrame indicando el índice de las filas y columnas que queremos acceder. Hay que tener en cuenta que los índices son valores enteros que comienzan con el cero, si empezamos a contar desde el primer elemento. Si empezamos a contar desde el último elemento, en sentido inverso, los índices comienzan en -1 y son negativos. La Figura 3 muestra ejemplos del uso de *loc* e *iloc*.

```
In [15]: tabla.loc[['Abr', 'Jul'], ['Temp.2021', 'Temp.2022']]

Out[15]:
```

	Temp.2021	Temp.2022
Abr	7	9
Jul	20	24

```
In [16]: tabla.iloc[[3,6],[-2, -1]]

Out[16]:
```

	Temp.2021	Temp.2022
Abr	7	9
Jul	20	24

Figura 3. Empleo de los indexadores *loc* e *iloc*.

Pandas también permite la selección de elementos usando *Series* de tipo bool (True o False) que actúan como filtro. Posteriormente, el filtro se aplica para indexar los elementos de una *Serie* o *DataFrame*. La Figura 4, muestra la creación de un filtro para seleccionar los meses con más de 30 días y su aplicación a una *Serie*. La Figura 5 y 6 muestran las generación y aplicación, respectivamente, de más ejemplos de filtros para acceder a datos de un *DataFrame*.

```
In [17]: 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')
filtro=dias>30
filtro

Out[17]: Ene    True
Feb    False
Mar    True
Abr    False
May    True
Jun    False
Jul    True
Ago    True
Sep    False
Oct    True
Nov    False
Dic    True
Name: Días de cada mes, dtype: bool

In [18]: meses_largos=dias[filtro]
meses_largos

Out[18]: Ene    31
Mar    31
May    31
Jul    31
Ago    31
Oct    31
Dic    31
Name: Días de cada mes, dtype: int64
```

Figura 4. Acceso a datos de una *Serie* mediante un filtro

```
In [24]: 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'])
filtro1=tabla['Estación']=='oto'
filtro1

Out[24]: Ene    False
Feb    False
Mar    False
Abr    False
May    False
Jun    False
Jul    False
Ago    False
Sep    False
Oct    True
Nov    True
Dic    True
Name: Estación, dtype: bool

In [25]: filtro2=tabla['Temp.2021']>10
filtro2

Out[25]: Ene    False
Feb    False
Mar    False
Abr    False
May    False
Jun    True
Jul    True
Ago    True
Sep    True
Oct    True
Nov    True
Dic    False
Name: Temp.2021, dtype: bool
```

Figura 5. Generación de filtros a partir de un *DataFrame*

In [26]:

tabla[filtro1]

Out[26]:

	Mes	Días	Estación	Temp.2021	Temp.2022	
	Oct	Octubre	31	oto	20	14
	Nov	Noviembre	30	oto	11	10
	Dic	Diciembre	31	oto	6	7

In [27]:

tabla[filtro2]

Out[27]:

	Mes	Días	Estación	Temp.2021	Temp.2022	
	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

In [28]:

tabla[filtro1 & filtro2]

Out[28]:

	Mes	Días	Estación	Temp.2021	Temp.2022	
	Oct	Octubre	31	oto	20	14
	Nov	Noviembre	30	oto	11	10

In [29]:

tabla[filtro1 | filtro2]

Out[29]:

	Mes	Días	Estación	Temp.2021	Temp.2022	
	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 6. Acceso a los datos de un Dataframe mediante filtros

## ACTUALIZACIÓN

Pandas permite eliminar filas y columnas, crear otras nuevas y modificar los valores de los datos contenidas en las ya existentes.

Es posible cambiar los valores de toda una columna, dando el mismo valor a todos sus elementos, o especificando el valor de cada uno de ellos. De igual modo es posible modificar los valores almacenados en filas y columnas concretas indexándolas apropiadamente, como se muestra en la Figura 7.

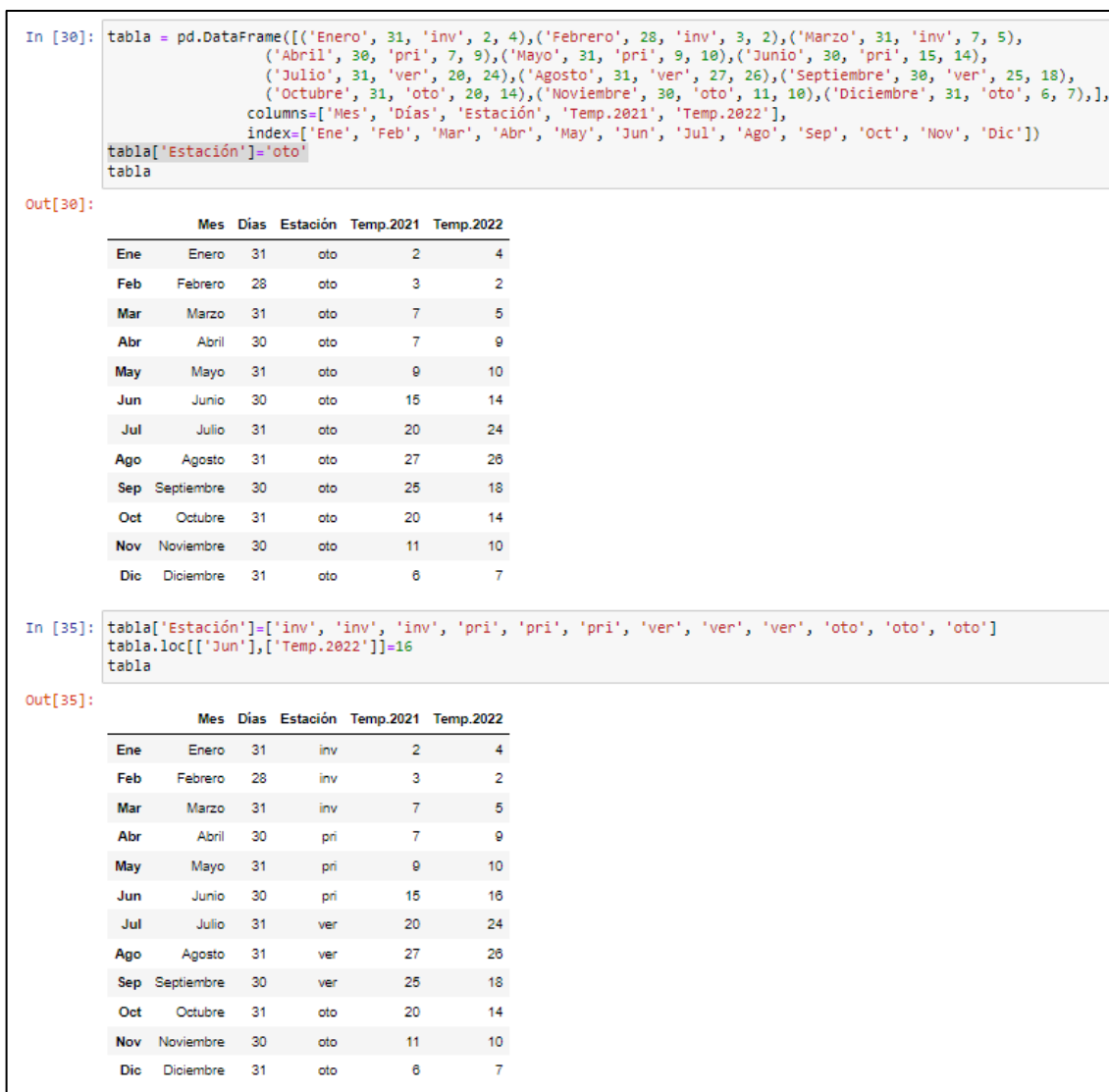


Figura 7. Actualización de datos de un DataFrame

Para eliminar datos de un DataFrame se pueden emplear las operaciones *pop* y *drop*, como muestran las Figuras 8 y 9. La primera permite eliminar una columna, modificando el *DataFrame*, y devuelve la columna eliminada en una estructura de tipo *Series*.

En cambio, *drop* permite eliminar tanto filas como columnas, pero no modifica el *DataFrame* original, sino que devuelve una copia, y no devuelve los datos eliminados. Si queremos que *drop* modifique el *DataFrame*, debemos usar el argumento *inplace* con valor True.

```
In [36]: tabla.pop('Temp.2022')
```

```
Out[36]: Ene    4
Feb     2
Mar     5
Abr     9
May    10
Jun    16
Jul    24
Ago    26
Sep    18
Oct    14
Nov    10
Dic     7
Name: Temp.2022, dtype: int64
```

```
In [37]: tabla
```

```
Out[37]:
```

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

Figura 8. Eliminación de datos con la operación pop

```
In [81]: nueva_tabla=tabla.drop(['Jul', 'Ago', 'Sep'])
nueva_tabla
```

```
Out[81]:
```

	Mes	Días	Estación	Temp.2021
Ene	Enero	31	inv	2
Feb	Febrero	28	inv	3
Mar	Marzo	31	inv	7
Abr	Abril	30	pri	7
May	Mayo	31	pri	9
Jun	Junio	30	pri	15
Oct	Octubre	31	oto	20
Nov	Noviembre	30	oto	11
Dic	Diciembre	31	oto	6

```
In [82]: tabla
```

```
Out[82]:
```

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

Figura 9. Eliminación de datos con la operación drop

Pandas también permite añadir filas (con el método *append*) y columnas a una tabla, como se muestra en la Figura 10.

In [83]: `nueva_tabla=nueva_tabla.append(pd.Series({'Mes':'Julio', 'Días': 31, 'Estación': 'ver', 'Temp.2021': 20}, name='Jul'))`  
`nueva_tabla=nueva_tabla.append(pd.Series({'Mes':'Agosto', 'Días': 31, 'Estación': 'ver', 'Temp.2021': 27}, name='Ago'))`  
`nueva_tabla=nueva_tabla.append(pd.Series({'Mes':'Septiembre', 'Días': 30, 'Estación': 'ver', 'Temp.2021': 25}, name='Sep'))`  
`nueva_tabla`

Out[83]:

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

In [84]: `nueva_tabla=nueva_tabla.reindex(['Ene', 'Feb', 'Mar', 'Abr', 'May', 'Jun', 'Jul', 'Ago', 'Sep', 'Oct', 'Nov', 'Dic'])`  
`nueva_tabla`

Out[84]:

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

In [85]: `nueva_tabla['Temp.2022']=[4,2,5,9,10,16,24,26,18,14,10,7]`  
`nueva_tabla`

Out[85]:

	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	16
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 10. Añadido de columnas y filas a un DataFrame



## OPERACIONES ARITMÉTICAS

Las *Series* y *DataFrames* proporcionan claridad en cuanto a los datos que contienen y su acceso, y sobre ellas pueden realizarse las funciones definidas en la librería *Numpy*, que permiten aplicar operaciones aritméticas (+, -, \*, /) entre otras.

Una característica muy importante en la realización de operaciones con Pandas es su capacidad para alinear dos estructuras (*Series* o *DataFrames*) de acuerdo con las etiquetas de sus índices. Esta capacidad es muy útil cuando se realizan operaciones entre estructuras que no tienen exactamente las mismas etiquetas en sus índices, o estos están desordenados, como en los ejemplos de la Figura 11.

```
In [88]: cosecha_finca1= pd.DataFrame([[300, 400], [100, 500], [400, 200], [600, 100]],
                                     columns=('trigo', 'maiz'), index=[2018, 2017, 2021, 2022])
cosecha_finca1

Out[88]:
```

	trigo	maiz
2018	300	400
2017	100	500
2021	400	200
2022	600	100

```
In [89]: cosecha_finca2= pd.DataFrame([[500, 300], [700, 400], [200, 800], [500, 600]],
                                     columns=('trigo', 'maiz'), index=[2017, 2018, 2022, 2020])
cosecha_finca2

Out[89]:
```

	trigo	maiz
2017	500	300
2018	700	400
2022	200	800
2020	500	600

```
In [92]: cosecha_total = cosecha_finca1 + cosecha_finca2
cosecha_total

Out[92]:
```

	trigo	maiz
2017	600.0	800.0
2018	1000.0	800.0
2020	NaN	NaN
2021	NaN	NaN
2022	800.0	900.0

```
In [93]: cosecha_finca1.add(cosecha_finca2, fill_value = 0)

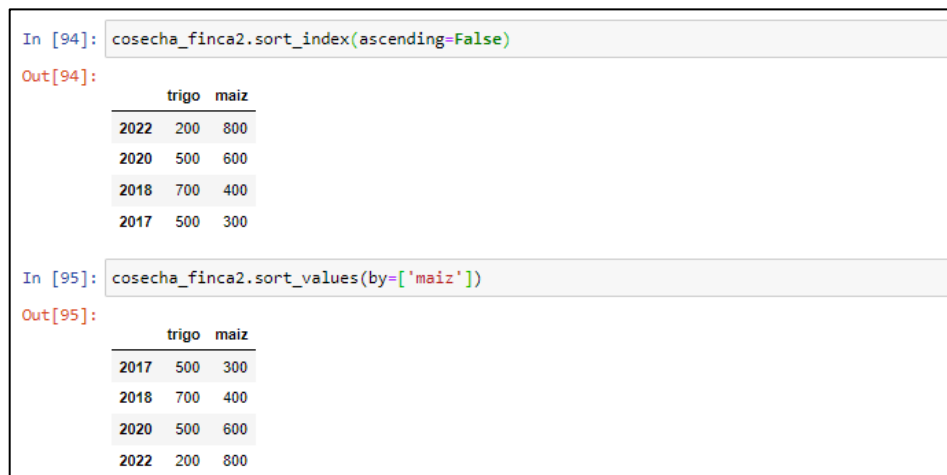
Out[93]:
```

	trigo	maiz
2017	600.0	800.0
2018	1000.0	800.0
2020	500.0	600.0
2021	400.0	200.0
2022	800.0	900.0

Figura 11. Ejemplos de suma de dos DataFrames

## ORDENACIÓN

Pandas ofrece los métodos `sort_index` y `sort_values` para ordenar, respectivamente, los índices y los valores de una Serie o DataFrame, como muestra la Figura 12. Estos métodos `sort_index` no modifican el DataFrame original, sino que devuelven una copia. Si queremos que modifiquen el DataFrame, hay que indicarlo dando el valor True al argumento `inplace`.



The screenshot shows two Jupyter Notebook cells. The first cell, labeled 'In [94]:', contains the code `cosecha_finca2.sort_index(ascending=False)`. The output, labeled 'Out[94]:', is a DataFrame with columns 'trigo' and 'maiz', sorted by index in descending order. The second cell, labeled 'In [95]:', contains the code `cosecha_finca2.sort_values(by=['maiz'])`. The output, labeled 'Out[95]:', is a DataFrame with columns 'trigo' and 'maiz', sorted by the 'maiz' column in ascending order.

```
In [94]: cosecha_finca2.sort_index(ascending=False)
Out[94]:
```

	trigo	maiz
2022	200	800
2020	500	600
2018	700	400
2017	500	300

```
In [95]: cosecha_finca2.sort_values(by=['maiz'])
Out[95]:
```

	trigo	maiz
2017	500	300
2018	700	400
2020	500	600
2022	200	800

Figura 12. Ordenación de índices y valores de un DataFrame

## 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.cienciadedatos.net/documentos/py06_machine_learning_python_scikitlearn.htm)

<https://www.drivendata.org/>