

1^a
Emisión

DATA SCIENCE

Módulo 05 Manipulación y visualización de datos con Python

Mtro. Ricardo Daniel Alanis Tamez



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
Dirección General de Cómputo y de Tecnologías de información y Comunicación
Dirección de Docencia en TIC



Educación
Continua
1971 - 2021

Transformaciones de datos

Manipulación y visualización de datos con Python

Ricardo Alanís

Presentación

En esta lección concluimos el trabajo previo de preparación de datos haciendo un tour por las distintas transformaciones de información.

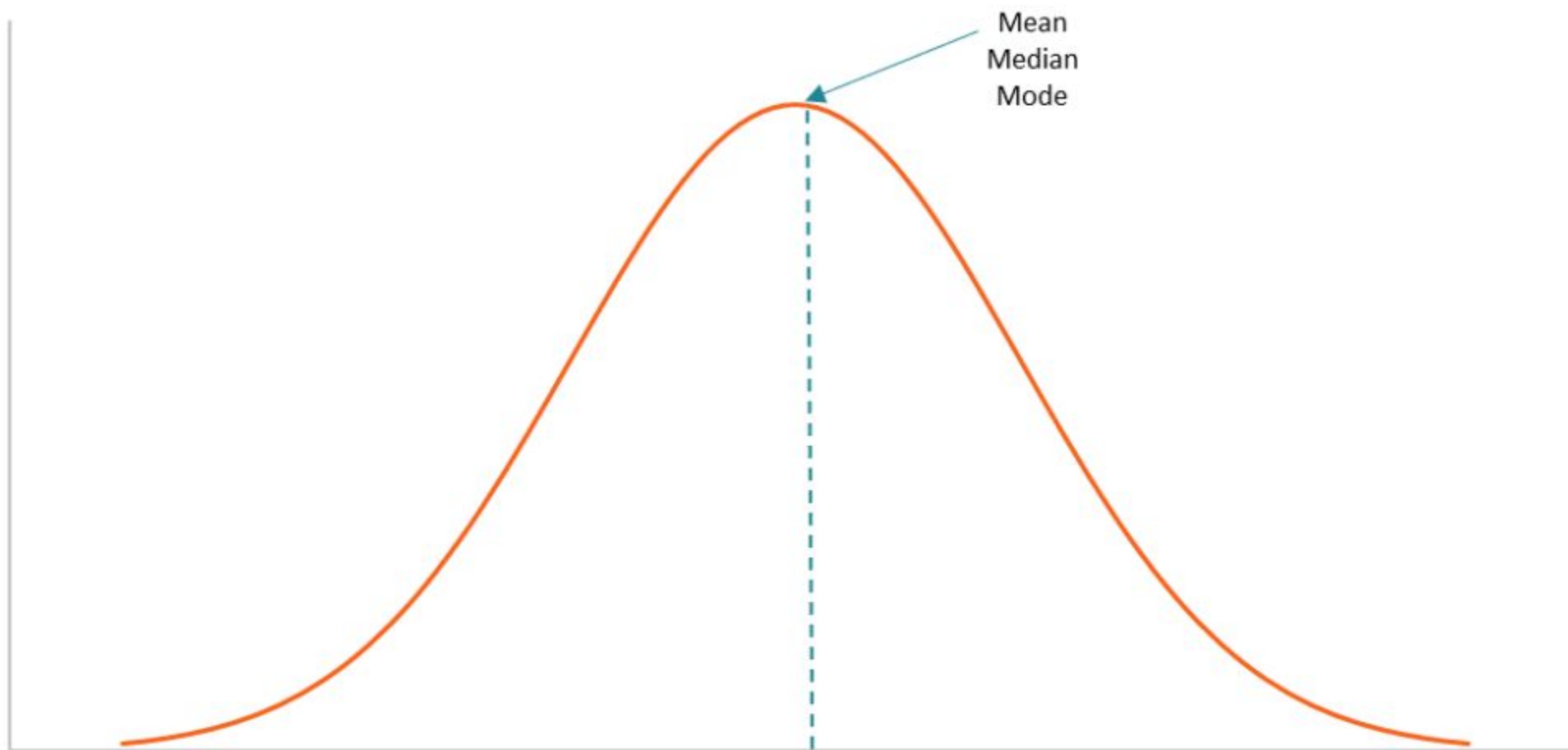
Objetivo

El participante identificará las diferentes transformaciones de datos.

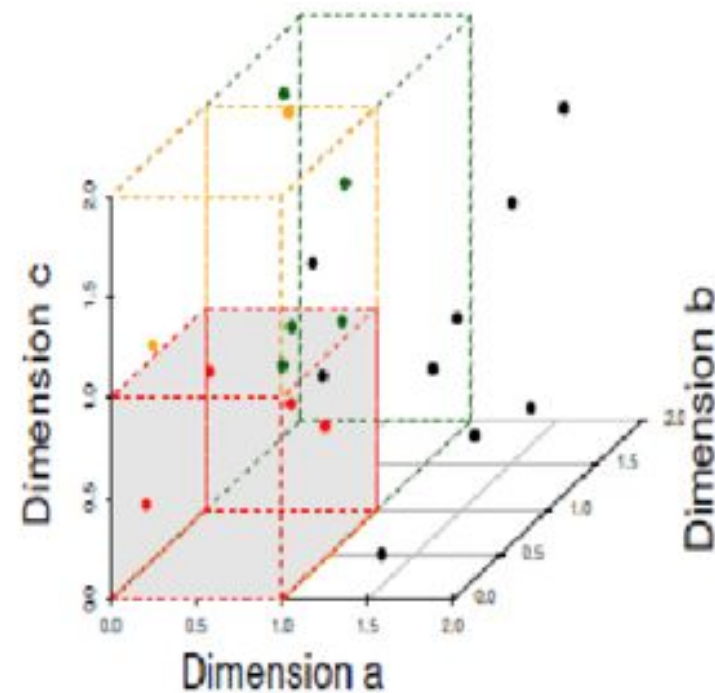
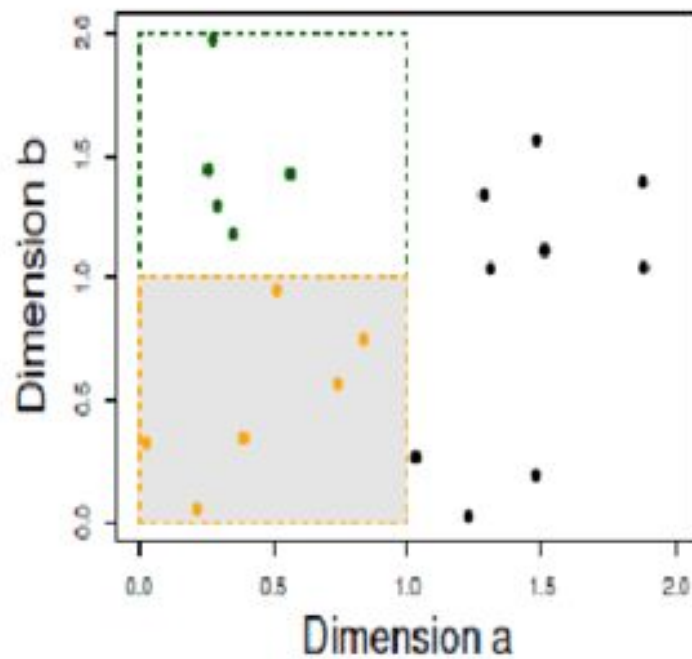
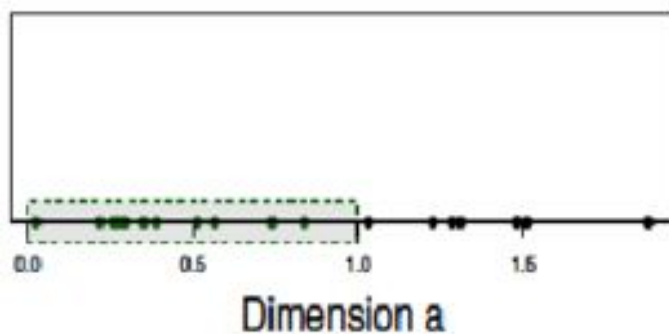
Agenda de hoy

- 1. Normalización**
- 2. Estandarización**
- 3. Codificación**
- 4. Binning**

Transformaciones de datos: Aspiración



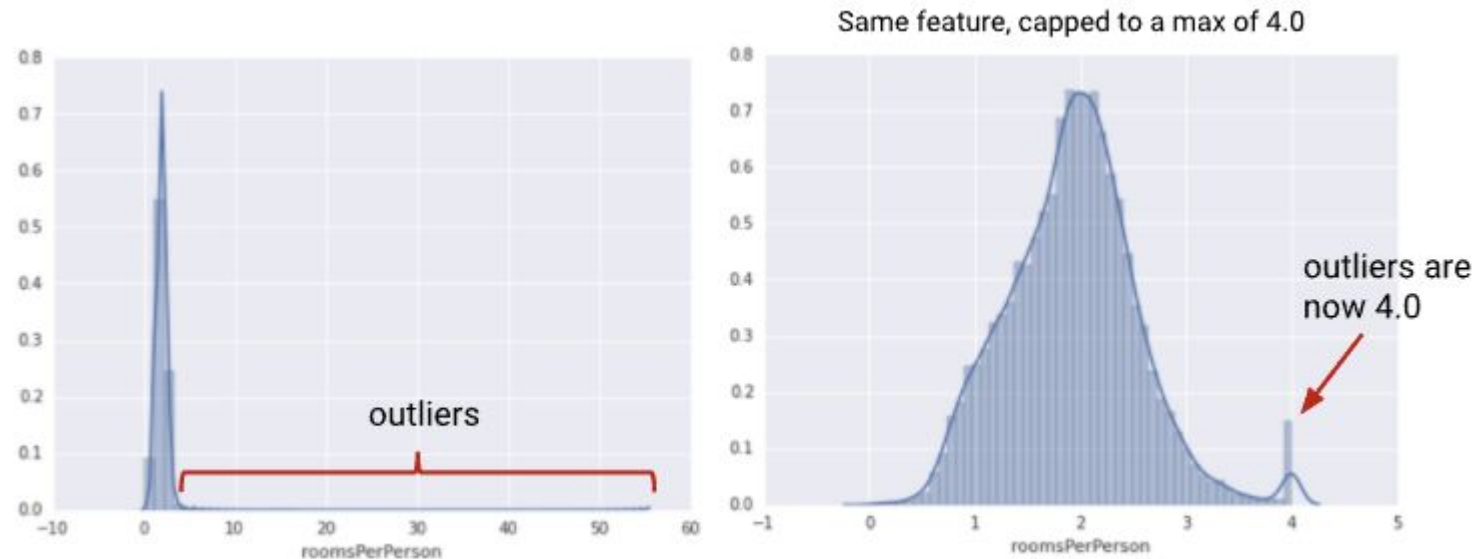
Qué pasa con las distancias entre datos



Normalización

Es la técnica de escala que mueve los valores y los reescala de manera que queden entre 0 y 1. También es conocido como Escala Min Max.

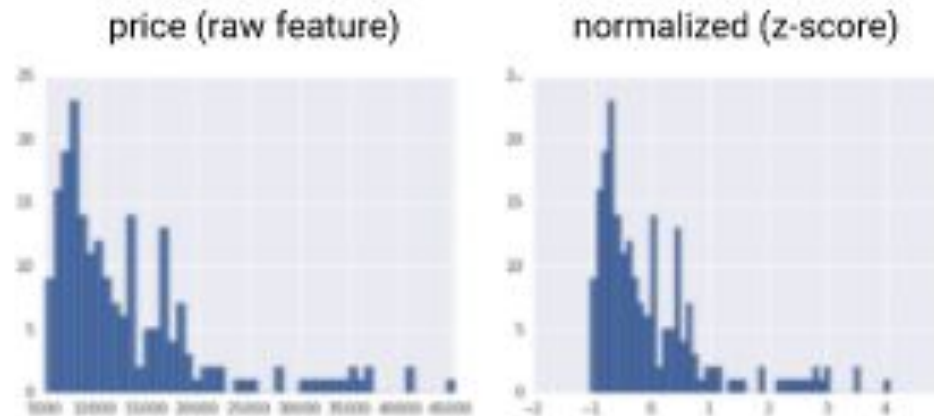
$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$



Estandarización

Es otra técnica de escala donde los valores están centrados entre la media y una desviación estándar. Esto significa que la media de los atributos se vuelve cero y la distribución tiene solo una unidad como desviación estándar

$$x' = (x - \mu) / \sigma$$



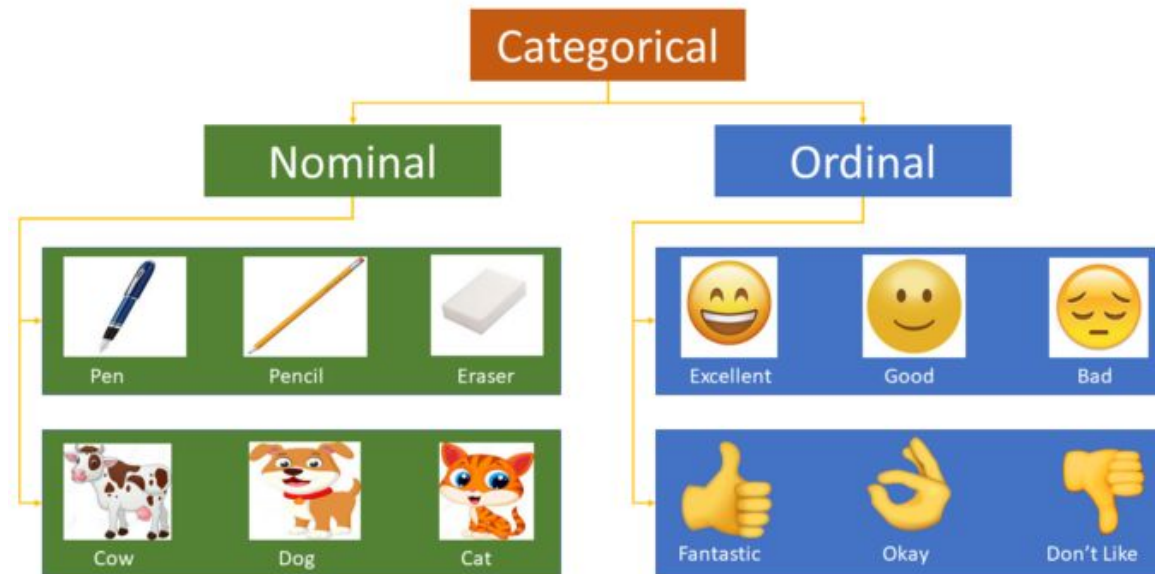
Ejercicio 6.1a: Manos a la obra

Normalización y Estandarización

- A mano
- Usando Pandas

Codificación


Es la estrategia de convertir variables de tipo texto a variables numéricas. Pueden ser del tipo nominal o categóricas, es decir, que importa o no su orden o escala.



Codificación: One hot

En este método, asignamos cada categoría a un vector que contiene 1 y 0 que denota la presencia de la característica o no. El número de vectores depende de las categorías que queramos mantener. Para características de alta cardinalidad, este método produce muchas columnas que ralentizan significativamente el aprendizaje.

User	City
1	Roma
2	Madrid
1	Madrid
3	Istanbul
2	Istanbul
1	Istanbul
1	Roma



User	Istanbul	Madrid
1	0	0
2	0	1
1	0	1
3	1	0
2	1	0
1	1	0
1	0	0

Ejercicio 6.1b: Manos a la obra

One hot

- A mano
- Usando Pandas

Media hora max

Codificación: Label - etiqueta

En esta codificación, a cada categoría se le asigna un valor de 1 a N (aquí N es el número de categoría para la característica). Puede verse como (Coche <Autobús <Camión... .0 <1 <2). Las categorías que tienen algunos vínculos o están cercanas entre sí pierden información después de la codificación.

CAT73	CAT73 label_encoded
A	1
A	1
C	3
B	2
A	1
C	3
B	2

Ejercicio 6.1c: Label encoding

Label encoding

- A mano
- Usando Pandas

Media hora max

Codificación: Frecuencia

Es una forma de utilizar la frecuencia de las categorías como etiquetas. En los casos en que la frecuencia está relacionada de alguna manera con la variable objetivo, ayuda al modelo a comprender y asignar el peso en proporción directa e inversa, según la naturaleza de los datos.

A	0.44 (4 out of 9)
B	0.33 (3 out of 9)
C	0.22 (2 out of 9)

Feature	Encoded Feature
A	0.44
A	0.44
A	0.44
A	0.44
B	0.33
B	0.33
B	0.33
C	0.22
C	0.22

Ejercicio 6.1d: Frequency

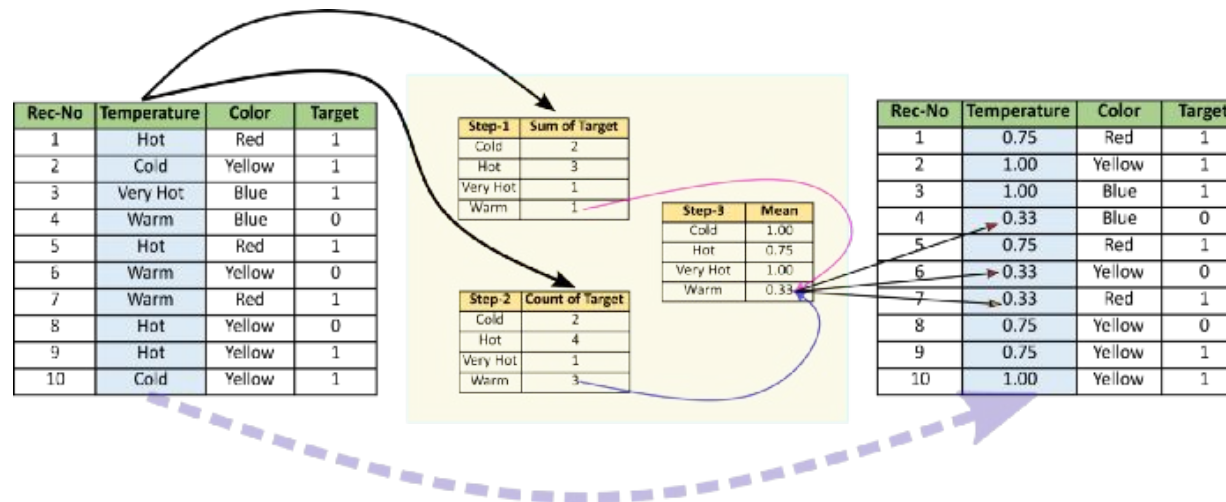
Manos a la obra

- Implementa una feature de frecuencia en el dataset de prueba

Media hora max

Codificación: Promedio (y otras aggs)

Mean Encoding or Target Encoding is one very popular encoding approach followed by Kagglers. Mean encoding is similar to label encoding, except here labels are correlated directly with the target. For example, in mean target encoding for each category in the feature label is decided with the mean value of the target variable on a training data.



Ejercicio 6.1e: Mean

Manos a la obra

- Implementa una feature de mean encoding en el dataset de prueba
- Utiliza cualquier otro tipo de agregación para generar un dataset

Media hora max

Binning

Binning se refiere al método tomará una columna con números continuos y colocará los números en "contenedores" según los rangos que determinemos. Esto nos dará una nueva característica de variable categórica.

	Car	MPG
0	BMW Z4 sDrive30i	28
1	Toyota Corolla Hybrid	52
2	Bugatti Chiron Pure Sport	10
3	Chevrolet Spark	33
4	Toyota Prius Eco	56
5	Volvo V90 FWD	26
6	Ford Mustang GT Premium (V8)	19
7	Mazda CX-9 Touring	22
8	Nissan Armada	14
9	Jeep Gladiator Sport	18



	Car	MPG	MPG_effeciency
0	BMW Z4 sDrive30i	28	Average Fuel Efficiency
1	Toyota Corolla Hybrid	52	Fuel Effecient
2	Bugatti Chiron Pure Sport	10	Gas Guzzler
3	Chevrolet Spark	33	Average Fuel Efficiency
4	Toyota Prius Eco	56	Fuel Effecient
5	Volvo V90 FWD	26	Average Fuel Efficiency
6	Ford Mustang GT Premium (V8)	19	Gas Guzzler
7	Mazda CX-9 Touring	22	Gas Guzzler
8	Nissan Armada	14	Gas Guzzler
9	Jeep Gladiator Sport	18	Gas Guzzler

Ejercicio 6.1f: Binning

Manos a la obra

- Implementa una feature de binning en el dataset de prueba

Media hora max

Join, Merge, Concatenate

Merge, join, concatenate and compare

pandas provides various facilities for easily combining together Series or DataFrame with various kinds of set logic for the indexes and relational algebra functionality in the case of join / merge-type operations.

In addition, pandas also provides utilities to compare two Series or DataFrame and summarize their differences.

Concatenating objects

The `concat()` function (in the main pandas namespace) does all of the heavy lifting of performing concatenation operations along an axis while performing optional set logic (union or intersection) of the indexes (if any) on the other axes. Note that I say "if any" because there is only a single possible axis of concatenation for Series.

Before diving into all of the details of `concat` and what it can do, here is a simple example:

```
In [1]: df1 = pd.DataFrame(  
...:     {  
...:         "A": ["A0", "A1", "A2", "A3"],  
...:         "B": ["B0", "B1", "B2", "B3"]  
...:     })
```

Concatenate

```
In [4]: frames = [df1, df2, df3]
```

```
In [5]: result = pd.concat(frames)
```

df1				
	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

df2				
	A	B	C	D
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7

df3				
	A	B	C	D
8	A8	B8	C8	D8
9	A9	B9	C9	D9
10	A10	B10	C10	D10
11	A11	B11	C11	D11

Result				
	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7
8	A8	B8	C8	D8
9	A9	B9	C9	D9
10	A10	B10	C10	D10
11	A11	B11	C11	D11

Merge/Join

combination.

```
In [39]: left = pd.DataFrame(  
.....:     {  
.....:         "key": ["K0", "K1", "K2", "K3"],  
.....:         "A": ["A0", "A1", "A2", "A3"],  
.....:         "B": ["B0", "B1", "B2", "B3"],  
.....:     }  
.....: )  
.....:  
  
In [40]: right = pd.DataFrame(  
.....:     {  
.....:         "key": ["K0", "K1", "K2", "K3"],  
.....:         "C": ["C0", "C1", "C2", "C3"],  
.....:         "D": ["D0", "D1", "D2", "D3"],  
.....:     }  
.....: )  
.....:  
  
In [41]: result = pd.merge(left, right, on="key")
```

left				right				Result					
	key	A	B		key	C	D		key	A	B	C	D
0	K0	A0	B0	0	K0	C0	D0	0	K0	A0	B0	C0	D0
1	K1	A1	B1	1	K1	C1	D1	1	K1	A1	B1	C1	D1
2	K2	A2	B2	2	K2	C2	D2	2	K2	A2	B2	C2	D2
3	K3	A3	B3	3	K3	C3	D3	3	K3	A3	B3	C3	D3

Compare

Comparing objects

The `compare()` and `compare()` methods allow you to compare two DataFrame or Series, respectively, and summarize their differences.

This feature was added in V1.1.0.

For example, you might want to compare two DataFrame and stack their differences side by side.

```
In [145]: df = pd.DataFrame(  
.....:     {  
.....:         "col1": ["a", "a", "b", "b", "a"],  
.....:         "col2": [1.0, 2.0, 3.0, np.nan, 5.0],  
.....:         "col3": [1.0, 2.0, 3.0, 4.0, 5.0],  
.....:     },  
.....:     columns=["col1", "col2", "col3"],  
.....: )  
.....:
```

```
In [146]: df  
Out[146]:  
   col1  col2  col3  
0     a   1.0   1.0  
1     a   2.0   2.0  
2     b   3.0   3.0  
3     b   NaN   4.0  
4     a   5.0   5.0
```

Ejercicio 6.1g: Merge, Join, Compare

Manos a la obra

- Implementa el join con el conjunto de datos de countries.
- Implementa un merge
- Separa los datos usando indexing y concatenarlo de nuevo

Media hora max

Práctica 6.1: Manos a la obra, para ti

Para el conjunto de datos de [jugadores](#) y [equipos](#):

- Implementa el merge/join para juntar los datos
- Implementa al menos 2 técnicas de filtrado o selección
- Encuentra 3 insights a compartir
- Prepara una presentación de 3 slides, una de cada etapa de tu trabajo

**Te debe de tomar
máximo 2 horas**

¿Preguntas?

Referencias

- Sivek, Susan., “Data Science 101: Normalization, Standardization, and Regularization.” KDnuggets. Apr. 2021.
www.kdnuggets.com/2021/04/data-science-101-normalization-standardization-regularization.html
- Tripathi, Himanshu., “Different Type of Feature Engineering Encoding Techniques for Categorical Variable Encoding.” Analytics Vidhya. 20 Sep, 2019.
medium.com/analytics-vidhya/different-type-of-feature-engineering-encoding-techniques-for-categorical-variable-encoding-214363a016fb
- Lutes, Jeremiah., “Binning for Feature Engineering in Machine Learning.” towards data science. 8 Jan, 2021.
towardsdatascience.com/binning-for-feature-engineering-in-machine-learning-d3b3d76f364a

Contacto

Mtro. Ricardo Daniel Alanis Tamez

ricardo@codeandomexico.org

LinkedIn: Ricardo Alanís