Eliminar valores perdidos o anómalos

En muchas ocasiones hay datos que no se han podido obtener. Ya sea porque el proceso de medida haya fallado, porque se han perdido en la comunicación o por cualquier otra razón, muchas veces nos encontramos con datos que faltan marcados como nulos. En otras ocasiones somos nosotros los que al detectar un valor anómalo lo marcamos como nulo.

Si nuestros datos contienen valores marcados como nulos (ya sea con `np.nan`, `None`, un `-1` en características que son sólo positivas o cualquier otra marca) y el algoritmo de aprendizaje no puede trabajar con ellos, podemos seguir varias estrategias:

- Eliminar los objetos (filas) con valores nulos
- Eliminar las caracteristicas (columnas) con valores nulos
- Sustituir los valores nulos por valores que estimemos adecuados

En este tema vamos a ver las dos primeras y la tercera se verá en el tema 2.5.

Valores atípicos como valores nulos

Como ejemplo, vamos a suponer que queremos descartar los valores anómalos detectados en el mismo conjunto de ejemplo del tema 1.6.

In [131]:
```
import pandas as pd
import numpy as np
from sklearn import datasets
dataset = datasets.fetch_openml(name='plasma_retinol', version=2, as_frame=True)
tabla = dataset.frame
tabla
```

Out[131]:

	AGE	SEX	SMOKSTAT	QUETELET	VITUSE	CALORIES	FAT	FIBER	ALCOHOL	CHOLESTEROL	BET
0	64.0	Female	Former	21.48380	Yes_fairly_often	1298.8	57.0	6.3	0.0	170.3	194
1	76.0	Female	Never	23.87631	Yes_fairly_often	1032.5	50.1	15.8	0.0	75.8	265
2	38.0	Female	Former	20.01080	Yes_not_often	2372.3	83.6	19.1	14.1	257.9	632
3	40.0	Female	Former	25.14062	No	2449.5	97.5	26.5	0.5	332.6	106
4	72.0	Female	Never	20.98504	Yes_fairly_often	1952.1	82.6	16.2	0.0	170.8	286
...	...	...	...	...	...	...	...	...	...	...	...
310	46.0	Female	Former	25.89669	No	2263.6	98.2	19.4	2.6	306.5	257
311	45.0	Female	Never	23.82703	Yes_fairly_often	1841.1	84.2	14.1	2.2	257.7	166
312	49.0	Female	Never	24.26126	Yes_fairly_often	1125.6	44.8	11.9	4.0	150.5	694
313	31.0	Female	Former	23.45255	Yes_fairly_often	2729.6	144.4	13.2	2.2	381.8	741
314	45.0	Female	Never	26.50808	Yes_fairly_often	1627.0	77.4	9.9	0.2	195.6	124

315 rows × 14 columns

En el siguiente código hacemos lo mismo que en el ejemplo del tema 1.6 pero, en vez de imprimir los valores que se salen del rango, los marcamos cambiandolos por `np.nan` (https://numpy.org/doc/stable/reference/constants.html#numpy.nan) (*Not a Number*) que es el valor de Numpy para los valores nulos. Nota: se puede ver escrito como `np.NAN` o `np.NaN` en algunos códigos. Son equivalentes pero la forma recomendada (https://numpy.org/doc/stable/reference/constants.html#numpy.NaN) es `np.nan`.

Además, vamos a anotar las columnas en las que aparecen los valores atípicos que hemos marcado como nulos (las filas ya las anotabamos). Como las columnas se van a visitar varias veces para evitar duplicados de forma eficiente usaremos la estructura de datos `set` (https://docs.python.org/3/library/stdtypes.html#set) de Python.

In [132]:
```
umbral_z_score = 4.0

# Calcular z-score, marcando atípicos con np.nan
filas_atipicos = []
columnas_atipicos = set()
desc = tabla.describe()
for i, fila in tabla.iterrows():
    atipico_detectado = False
    for caract in desc:
        z_score = abs(tabla.loc[i][caract] - desc.loc['mean'][caract]) / desc.loc['std'][carac
t]
        if z_score > umbral_z_score:
            atipico_detectado = True
            columnas_atipicos.add(caract)
            tabla.loc[i,caract] = np.nan
    if atipico_detectado:
        filas_atipicos.append(i)

filas_atipicos
```

Out[132]:
```
[39, 50, 61, 93, 151, 170, 207, 218, 225, 256, 261, 262, 308]
```

In [133]:
```
columnas_atipicos
```

Out[133]:
```
{'ALCOHOL',
 'BETADIET',
 'BETAPLASMA',
 'CALORIES',
 'CHOLESTEROL',
 'FAT',
 'FIBER',
 'QUETELET',
 'RETDIET'}
```

Podemos ver por ejemplo los cuatro valores nulos de las filas 256, 261 y 262 que Pandas muestra como 'NaN':

In [134]:
```
tabla.iloc[256:263]
```

Out[134]:

	AGE	SEX	SMOKSTAT	QUETELET	VITUSE	CALORIES	FAT	FIBER	ALCOHOL	CHOLESTEROL	BET
256	40.0	Female	Never	31.24219	Yes_fairly_often	3014.9	165.7	14.4	0.0	NaN	102
257	29.0	Female	Never	37.93996	Yes_fairly_often	1631.0	55.6	13.8	0.5	189.5	343
258	71.0	Female	Former	24.98825	No	1399.5	66.5	9.6	8.0	260.0	152
259	45.0	Female	Never	23.43164	Yes_fairly_often	2319.0	122.1	13.4	0.1	305.7	204
260	63.0	Female	Never	18.92094	No	1655.9	70.8	15.1	0.1	177.3	289
261	46.0	Female	Former	24.26126	Yes_not_often	1422.8	58.3	7.8	7.1	206.3	198
262	75.0	Female	Never	21.67837	Yes_fairly_often	2511.5	92.3	NaN	0.6	228.3	427

Como en el código hemos extraído las filas en las que encontramos valores atípicos, podemos eliminarla directamente usando el método drop (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.drop.html):

In [135]:
```
tabla_sin_filas_atipicas = tabla.drop(filas_atipicos, axis=0)
tabla_sin_filas_atipicas
```

Out[135]:

	AGE	SEX	SMOKSTAT	QUETELET	VITUSE	CALORIES	FAT	FIBER	ALCOHOL	CHOLESTEROL	BET
0	64.0	Female	Former	21.48380	Yes_fairly_often	1298.8	57.0	6.3	0.0	170.3	194
1	76.0	Female	Never	23.87631	Yes_fairly_often	1032.5	50.1	15.8	0.0	75.8	265
2	38.0	Female	Former	20.01080	Yes_not_often	2372.3	83.6	19.1	14.1	257.9	632
3	40.0	Female	Former	25.14062	No	2449.5	97.5	26.5	0.5	332.6	106
4	72.0	Female	Never	20.98504	Yes_fairly_often	1952.1	82.6	16.2	0.0	170.8	286
...	...	...	...	...	...	...	...	...	...	...	...
310	46.0	Female	Former	25.89669	No	2263.6	98.2	19.4	2.6	306.5	257
311	45.0	Female	Never	23.82703	Yes_fairly_often	1841.1	84.2	14.1	2.2	257.7	166
312	49.0	Female	Never	24.26126	Yes_fairly_often	1125.6	44.8	11.9	4.0	150.5	694
313	31.0	Female	Former	23.45255	Yes_fairly_often	2729.6	144.4	13.2	2.2	381.8	741
314	45.0	Female	Never	26.50808	Yes_fairly_often	1627.0	77.4	9.9	0.2	195.6	124

302 rows × 14 columns

Y de la misma forma, podríamos eliminar las columnas.

In [136]:
```
tabla_sin_columnas_atipicas = tabla.drop(columnas_atipicos, axis=1)
tabla_sin_columnas_atipicas
```

Out[136]:

	AGE	SEX	SMOKSTAT	VITUSE	binaryClass
0	64.0	Female	Former	Yes_fairly_often	N
1	76.0	Female	Never	Yes_fairly_often	N
2	38.0	Female	Former	Yes_not_often	N
3	40.0	Female	Former	No	N
4	72.0	Female	Never	Yes_fairly_often	N
...	...	...	...	...	...
310	46.0	Female	Former	No	P
311	45.0	Female	Never	Yes_fairly_often	P
312	49.0	Female	Never	Yes_fairly_often	P
313	31.0	Female	Former	Yes_fairly_often	N
314	45.0	Female	Never	Yes_fairly_often	N

315 rows × 5 columns

Como veis, eliminar las columnas suele ser mucho más drástico que eliminar las filas con valores nulos. En este caso perdemos la mitad de la información. Mientras que, eliminando las filas sólo se perdía un 4% ($\frac{13}{315}$) de los datos. No obstante, puede haber casos donde pocas columnas sean las problemáticas, por ejemplo, porque sean características más difíciles de medir. En esos casos puede que no sean características interesantes para el procesado posterior y merezca la pena probar sin ellas.

Conjuntos de datos con valores perdidos

Nos podemos encontrar con conjuntos de dato que tienen algunos valores marcados como nulos, de la misma forma que hicimos nosotros en el caso anterior pero, en ese caso, no tenemos la lista de filas o columnas con valores nulos.

In [137]:
```
dataset = datasets.fetch_openml(name='sleep', version=2, as_frame=True)
tabla_sleep = dataset.frame
tabla_sleep
```

Out[137]:

	body_weight	brain_weight	slow_wave	paradoxical	total_sleep	maximum_life_span	gestation_time	predation_ir
0	6654.000	5712.0	NaN	NaN	3.3	38.6	645.0	3.0
1	1.000	6.6	6.3	2.0	8.3	4.5	42.0	3.0
2	3.385	44.5	NaN	NaN	12.5	14.0	60.0	1.0
3	0.920	5.7	NaN	NaN	16.5	NaN	25.0	5.0
4	2547.000	4603.0	2.1	1.8	3.9	69.0	624.0	3.0
...	...	...	...	...	...	...	...	...
57	2.000	12.3	4.9	0.5	5.4	7.5	200.0	3.0
58	0.104	2.5	13.2	2.6	15.8	2.3	46.0	3.0
59	4.190	58.0	9.7	0.6	10.3	24.0	210.0	4.0
60	3.500	3.9	12.8	6.6	19.4	3.0	14.0	2.0
61	4.050	17.0	NaN	NaN	NaN	13.0	38.0	3.0

62 rows × 10 columns

Ejercicio: Usando la función `pd.isnull()` de Pandas (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.isnull.html), crear las listas de filas y columnas que tienen valores nulos en el conjunto de datos anterior.

Ahora que ya tienes un buen dominio recorriendo la tabla marcando e identificando los valores nulos, es un buen momento para saber que hay una función en Pandas para eliminar los valores marcados como nulos directamente. Es interesante conocer lo anterior porque muchas veces uno se encuentra con datos donde los valores nulos están marcados con otras cosas como la cadena 'NULL' o un -1. En ese caso sería fácil modificar el código anterior para detectarlos haciendo un código 'a medida'.

Nuestra recomendación en esos casos es que se haga una transformación de esos valores al identificador estándar de Pandas (`np.nan` en características numéricas, None en objetos, NaT en fechas) junto con los procesos de estructurar los datos que cometabamos al final del tema 3. De esta forma, cuando vemos como tratar con los valores nulos, ya tenemos eso bien ordenado y nos podemos centrar en estudiar la mejor estrategia para ellos (eliminarlos o sustituirlos).

La método de los `DataFrame` de Pandas para elimnar valores considerados nulos es: `dropna` (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.dropna.html#pandas.DataFrame.dropna).

Ejercicio: Leer la documentación de `dropna` (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.dropna.html#pandas.DataFrame.dropna) y averiguar lo que hace el siguiente ejemplo:

In [138]:
```
tabla_sleep_sin_nulos = tabla_sleep.dropna(axis=0, thresh=8)
tabla_sleep_sin_nulos
```

Out[138]:

	body_weight	brain_weight	slow_wave	paradoxical	total_sleep	maximum_life_span	gestation_time	predation_ir
0	6654.000	5712.00	NaN	NaN	3.3	38.6	645.0	3.0
1	1.000	6.60	6.3	2.0	8.3	4.5	42.0	3.0
2	3.385	44.50	NaN	NaN	12.5	14.0	60.0	1.0
4	2547.000	4603.00	2.1	1.8	3.9	69.0	624.0	3.0
5	10.550	179.50	9.1	0.7	9.8	27.0	180.0	4.0
6	0.023	0.30	15.8	3.9	19.7	19.0	35.0	1.0
7	160.000	169.00	5.2	1.0	6.2	30.4	392.0	4.0
8	3.300	25.60	10.9	3.6	14.5	28.0	63.0	1.0
9	52.160	440.00	8.3	1.4	9.7	50.0	230.0	1.0
10	0.425	6.40	11.0	1.5	12.5	7.0	112.0	5.0
11	465.000	423.00	3.2	0.7	3.9	30.0	281.0	5.0
12	0.550	2.40	7.6	2.7	10.3	NaN	NaN	2.0
13	187.100	419.00	NaN	NaN	3.1	40.0	365.0	5.0
14	0.075	1.20	6.3	2.1	8.4	3.5	42.0	1.0
15	3.000	25.00	8.6	0.0	8.6	50.0	28.0	2.0
16	0.785	3.50	6.6	4.1	10.7	6.0	42.0	2.0
17	0.200	5.00	9.5	1.2	10.7	10.4	120.0	2.0
18	1.410	17.50	4.8	1.3	6.1	34.0	NaN	1.0
19	60.000	81.00	12.0	6.1	18.1	7.0	NaN	1.0
20	529.000	680.00	NaN	0.3	NaN	28.0	400.0	5.0
21	27.660	115.00	3.3	0.5	3.8	20.0	148.0	5.0
22	0.120	1.00	11.0	3.4	14.4	3.9	16.0	3.0
23	207.000	406.00	NaN	NaN	12.0	39.3	252.0	1.0
24	85.000	325.00	4.7	1.5	6.2	41.0	310.0	1.0
25	36.330	119.50	NaN	NaN	13.0	16.2	63.0	1.0
26	0.101	4.00	10.4	3.4	13.8	9.0	28.0	5.0
27	1.040	5.50	7.4	0.8	8.2	7.6	68.0	5.0
28	521.000	655.00	2.1	0.8	2.9	46.0	336.0	5.0
29	100.000	157.00	NaN	NaN	10.8	22.4	100.0	1.0
31	0.005	0.14	7.7	1.4	9.1	2.6	21.5	5.0
32	0.010	0.25	17.9	2.0	19.9	24.0	50.0	1.0
33	62.000	1320.00	6.1	1.9	8.0	100.0	267.0	1.0
34	0.122	3.00	8.2	2.4	10.6	NaN	30.0	2.0
35	1.350	8.10	8.4	2.8	11.2	NaN	45.0	3.0
36	0.023	0.40	11.9	1.3	13.2	3.2	19.0	4.0
37	0.048	0.33	10.8	2.0	12.8	2.0	30.0	4.0
38	1.700	6.30	13.8	5.6	19.4	5.0	12.0	2.0
39	3.500	10.80	14.3	3.1	17.4	6.5	120.0	2.0
40	250.000	490.00	NaN	1.0	NaN	23.6	440.0	5.0
41	0.480	15.50	15.2	1.8	17.0	12.0	140.0	2.0
42	10.000	115.00	10.0	0.9	10.9	20.2	170.0	4.0
43	1.620	11.40	11.9	1.8	13.7	13.0	17.0	2.0
44	192.000	180.00	6.5	1.9	8.4	27.0	115.0	4.0
45	2.500	12.10	7.5	0.9	8.4	18.0	31.0	5.0

	body_weight	brain_weight	slow_wave	paradoxical	total_sleep	maximum_life_span	gestation_time	predation_i
46	4.288	39.20	NaN	NaN	12.5	13.7	63.0	2.0
47	0.280	1.90	10.6	2.6	13.2	4.7	21.0	3.0
48	4.235	50.40	7.4	2.4	9.8	9.8	52.0	1.0
49	6.800	179.00	8.4	1.2	9.6	29.0	164.0	2.0
50	0.750	12.30	5.7	0.9	6.6	7.0	225.0	2.0
51	3.600	21.00	4.9	0.5	5.4	6.0	225.0	3.0
52	14.830	98.20	NaN	NaN	2.6	17.0	150.0	5.0
53	55.500	175.00	3.2	0.6	3.8	20.0	151.0	5.0
54	1.400	12.50	NaN	NaN	11.0	12.7	90.0	2.0
55	0.060	1.00	8.1	2.2	10.3	3.5	NaN	3.0
56	0.900	2.60	11.0	2.3	13.3	4.5	60.0	2.0
57	2.000	12.30	4.9	0.5	5.4	7.5	200.0	3.0

Ejercicio: aplicar `dropna` para eliminar todas las filas con valores nulos.

Ejercicio: aplicar `dropna` para eliminar todas las columnas con valores nulos.