Relaciones entre características

Puede ser interesante conocer las relaciones entre las características de nuestros datos.

Relaciones entre variables numéricas

Vamos a ver un ejemplo sobre un problema de regresión con 5 variables.

In [78]:
```
import pandas as pd
import numpy as np
from sklearn import datasets
```

In [79]:
```
dataset = datasets.fetch_openml(name='SolarPower', as_frame=True)
tabla = dataset.frame
tabla
```

Out[79]:

	rhumidity	temperature	windspeed	solarirradiance	pvoutput
0	29.82	21.94	1.65	8.38	1.185
1	30.83	22.67	2.67	7.92	1.301
2	27.74	21.37	2.30	7.54	1.356
3	26.40	21.88	1.78	7.54	1.288
4	23.09	21.96	1.91	7.73	1.239
...	...	...	...	...	...
199	89.16	25.57	5.49	10.60	1.528
200	85.96	26.11	5.14	10.70	1.451
201	86.97	26.41	5.20	10.84	1.312
202	89.86	25.44	4.51	10.79	1.460
203	87.32	25.79	4.30	10.68	1.391

204 rows × 5 columns

Empecemos viendo como lo podríamos hacer en bajo nivel, si tenemos los datos en arrays de Numpy, podemos usar sus funciones para ver la relación entre variables. Lo más básico es la calcular la covarianza (https://es.wikipedia.org/wiki/Covarianza):

In [80]:
```
array_humedad = np.array(tabla['rhumidity'])
array_temperatura = np.array(tabla['temperature'])
np.cov(array_humedad, array_temperatura)
```

Out[80]:
```
array([[693.53137768, -55.90797155],
       [-55.90797155,  26.26752179]])
```

El resultado nos indica en la posición `[0,0]` la varianza de la humedad y en la `[1,1]` la varizanza de la temperatura. En las posiciones `[0,1]` y `[1,0]` nos da el valor de la convarianza que estabamos buscando entre ambas variables. Devuelve una matriz porque podríamos pasarle arrays con varias características, en vez de una como en este ejemplo, y nos calcularía la covarianza entre todas ellas. La matriz de resultado es simétrica porque la covarianza es conmutativa.

El valor -55.9 nos indica que la posible relación lineal entre temperatura y humedad (en estos datos) es inversa (cuanto mayor sea una, menor será la otra).

Sin embargo, la magnitud -55.9 por si sola no nos aporta una indicación de la intensidad de la relación porque depende de lo que varien cada una. Hay que observarla junto con las varianzas. Por ello, lo más habitual es medir la correlación lineal (https://es.wikipedia.org/wiki/Correlaci%C3%B3n) entre características. Si tenemos dos arrays de Numpy podemos usar la función `np.coorcoef`:

```
In [81]: np.corrcoef(array_humedad, array_temperatura)
```

```
Out[81]: array([[ 1.        , -0.41422006],
                [-0.41422006,  1.        ]])
```

En el resultado podemos ver la relación lineal inversa (a mayor humedad, menor temperatura) pero, esta vez, como los valores sabemos que están en el intervalo [-1, 1], sabemos que la relación es bastante fuerte.

Podemos intentar ver como de significativa es esta relación. ¿Cuál es la probabilidad de que estos valores hayan salido así aleatoriamente de dos variables que realmente son independientes (no tienen relación ninguna)?

```
In [82]: from scipy.stats.stats import pearsonr
         pearsonr(array_humedad, array_temperatura)
```

```
Out[82]: (-0.414220060261395, 7.330929500386973e-10)
```

El p-value es MUY pequeño. Esto indica que la probabilidad de que esa relación lineal no exista es muy baja. Podemos descartar que sean independientes y confirmar que hay alguna relación lineal que influencia los cambios de valores entre ambas (puede haber otros factores que también afecten a estas características).

Si estamos trabajando con Pandas, podemos calcular las correlaciones de forma similar, con la ventaja de que los resultados son más fáciles de leer al llevar los nombres de las características.

In [83]:
```
tabla.corr()
```

Out[83]:

	rhumidity	temperature	windspeed	solarirradiance	pvoutput
rhumidity	1.000000	-0.414220	0.355240	0.560150	0.099394
temperature	-0.414220	1.000000	0.106980	0.478715	-0.109598
windspeed	0.355240	0.106980	1.000000	0.370385	0.123514
solarirradiance	0.560150	0.478715	0.370385	1.000000	-0.027745
pvoutput	0.099394	-0.109598	0.123514	-0.027745	1.000000

Relaciones entre variables categóricas

Una medida que puede ser muy útil en el contexto de variables categóricas es la medida de la información mútua (https://en.wikipedia.org/wiki/Mutual_information) basada en la teoría de la información (https://en.wikipedia.org/wiki/Information_theory). Vamos a utilizar otro conjunto de datos como ejemplo:

In [84]:
```
dataset = datasets.fetch_openml(name='qualitative-bankruptcy', as_
frame=True)
tabla = dataset.frame
tabla
```

Out[84]:

	V1	V2	V3	V4	V5	V6	Class
0	3	3	1	1	1	3	2
1	2	2	1	1	1	2	2
2	1	1	1	1	1	1	2
3	3	3	3	3	3	3	2
4	2	2	3	3	3	2	2
...	...	...	...	...	...	...	...
245	2	2	1	2	2	2	1
246	3	2	2	2	2	2	1
247	1	2	2	2	2	2	1
248	2	2	2	2	2	2	1
249	3	2	2	2	1	1	1

250 rows × 7 columns

Para calcular la información que aporta una variable respecto a otra podemos usar la función `mutual_info_score` del paquete sklearn. Así podemos ver la información que aporta 'V1' sobre 'V2' (y viceversa porque es simétrica):

In [85]:
```
from sklearn.metrics import mutual_info_score
mutual_info_score(tabla['V1'], tabla['V2'])
```

Out[85]: 0.09677929526063088

Probablemente, nos interesará más ver la información que aporta sobre la clase:

In [86]:
```
mutual_info_score(tabla['V1'], tabla['Class'])
```

Out[86]: 0.03183844900524929

Estos valores representan la cantidad de información medida en nats. Un nat equivale a 1.4427 bits. Aunque nos puede resultar más intuitivo el bit, la razón por la que el resultado está en nat es porque las funciones usan el logaritmo neperiano. No tiene mayor importancia ya que la conversión es muy sencilla.

In [87]:
```
import math
bits_per_nat = math.log(math.e,2)
bits_per_nat
```

Out[87]: 1.4426950408889634

Así que en bits serían:

In [88]:
```
print( mutual_info_score(tabla['V1'], tabla['V2']) * bits_per_nat,
'bits' )
print( mutual_info_score(tabla['V1'], tabla['Class']) * bits_per_n
at, 'bits')
```

```
0.13962300933324093 bits
0.0459331724894693 bits
```

Por otra parte, también es relativo a la cantidad de información que es necesaria para identificar completamente la otra característica. Eso es equivalente a la cantidad de entropia de la característica.

In [89]:
```
import scipy.stats
for col in tabla:
    print('{:>10} {:.3f}'.format(
            col,
            scipy.stats.entropy(tabla[col].value_counts() / len(ta
bla))*bits_per_nat))
```

```
        V1 1.583
        V2 1.521
        V3 1.516
        V4 1.579
        V5 1.541
        V6 1.528
     Class 0.985
```

La clase tiene 2 valores y, por tanto, necesita casi un bit para representarse. No llega al bit entero porque la probabilidad de uno de los valores es más alta que la otra. Así, en promedio, para almacenar una gran cantidad de valores de esa clase en un formato comprimido necesitaríamos algo menos de 1 bit por valor. Las otras variables tienen 3 valores y como sus valores están bastante balanceados, se aproximan a 1.59bits = $log_2(3)$.

Para poder ver todas las relaciones, como veíamos en la matriz de correlaciones, podemos programar nosotros la presentación y sacar también una tabla:

In [90]:
```
print(' '*10, end='')
for col in tabla:
    print('{:>6}'.format(col), end='')
print()
for col in tabla:
    print('{:>10}'.format(col), end=' ')
    for col2 in tabla:
        print('{:.3f}'.format(mutual_info_score(tabla[col], tabla
[col2]) * bits_per_nat), end=' ')
    print()
```

```
                V1    V2    V3    V4    V5    V6 Class
        V1 1.583 0.140 0.024 0.009 0.060 0.060 0.046
        V2 0.140 1.521 0.064 0.089 0.107 0.084 0.106
        V3 0.024 0.064 1.516 0.337 0.550 0.027 0.585
        V4 0.009 0.089 0.337 1.579 0.477 0.063 0.533
        V5 0.060 0.107 0.550 0.477 1.541 0.072 0.902
        V6 0.060 0.084 0.027 0.063 0.072 1.528 0.059
     Class 0.046 0.106 0.585 0.533 0.902 0.059 0.985
```

Ejercicio: Coge un par de conjuntos de datos a tu elección (datos propios, extraidos, o de alguno de los repositorios) y calcula las correlaciones entre variables continuas y la información mútua entre variables categóricas.