EDUCACIÓN FINANCIERA

MUTUAL INFORMATION: UNA HERRAMIENTA ANALÍTICA PARA CUANTIFICAR RELACIONES NO LINEALES

Esta columna introduce el concepto de *Mutual Information* para medir la cantidad de información que se puede obtener sobre una variable aleatoria mediante la observación de otra. Esta herramienta analítica tiene su origen en la "teoría de la información" y puede ser muy útil en análisis financiero para identificar relaciones no lineales entre variables aleatorias.

Por Francisco Barañao, CFA.

Una tarea habitual en el análisis matemático es cuantificar la relación entre dos variables aleatorias.

En el caso del análisis financiero, por ejemplo, podría ser necesario analizar la relación en el precio de una acción y el precio de un *commodity*. O entre el valor de un bono y su clasificación de riesgo.

La medida más utilizada para cuantificar estas relaciones es la correlación, media generalmente por el coeficiente de correlación de Pearson.

Esta medida tiene la ventaja de ser fácil de calcular, y es muy utilizada. Sin embargo, sólo mide correlaciones lineales.

La teoría de la información proporciona una medida alternativa que permite cuantificar la correlación entre dos variables cualquiera, sea esta lineal o no.

La teoría de la información fue desarrollada por Claude E. Shannon a partir de su famosa publicación "A Mathematical Theory of Communication", en 1948, donde introdujo por primera vez el modelo cualitativo y cuantitativo de la comunicación como un proceso estadístico subyacente a la teoría de la información

En esta columna solo vamos a explicar brevemente el concepto de "Mutual Information" (MI), dejando de lado los otros elementos de esta teoría, como su relación con el concepto de "entropía".

MI mide **la cantidad de información que se puede obtener sobre una variable aleatoria mediante la observación de otra**.

La fórmula de MI es la siguiente:

$$I(X,Y) = \sum_x \sum_y p(xy) \log \left(\frac{p(x,y)}{p(x)p(y)} \right)$$

Donde X e Y son variables aleatorias, *p(x,y)* es la probabilidad conjunta y *p(x)* y *p(y)* son las probabilidades marginales.

Tradicionalmente se utiliza el logaritmo en base 2, pero es posible utilizar logaritmos en cualquier base.

En primer lugar, vamos a explicar el cálculo detallado para un ejemplo muy simple con variables discretas[1].

En este ejemplo, supongamos que un analista de crédito ha identificado una característica de los deudores (variable X, o "*feature*") que puede tomar 3 valores discretos. Por ejemplo, tener un historial de crédito bueno, malo o inexistente.

[1] Este ejemplo está basado en "Lecture 2: Entropy and Mutual Information", Dr. Yao Xie, ECE587, Information Theory, Duke University https://www2.isye.gatech.edu/~yxie77/ece587/Lecture2.pdf

Esta variable X se asocia con 3 tipos distintos de incumplimiento (variable Y, o "*target*"). Por ejemplo: atraso temporal en el pago, no pago de la deuda, o desaparición del deudor.

A partir de una tabla con datos históricos se han identificado las siguientes frecuencias entre cada una de las características e incumplimientos:

		Características (X)			**Total**
		1	**2**	**3**	
Incumplimientos (Y)	**1**	3	2	0	**5**
	2	1	0	4	**5**
	3	4	0	4	**8**
Total		**8**	**2**	**8**	**18**

Es decir: de 18 casos, en 3 de ellos se observó en el deudor la característica 1 y el incumplimiento 1, mientras que en 2 de ellos se observó la característica 2 y el incumplimiento 1.

Para calcular la MI entre ambas variables se construye una matriz con las probabilidades conjuntas para cada caso dentro de la matriz, y las probabilidades marginales en la fila inferior y en la de la derecha:

		Características (X)			
		1	**2**	**3**	**Total**
Incumplimientos (Y)	**1**	0,17	0,11	0,00	**0,28**
	2	0,06	0,00	0,22	**0,28**
	3	0,22	0,00	0,22	**0,44**
Total		**0,44**	**0,11**	**0,44**	**1,00**

Luego se aplica la fórmula para uno de los 9 casos de la matriz, donde p(x,y) corresponde a la probabilidad conjunta y p(x) y p(y) corresponden a las probabilidades marginales. Si la probabilidad conjunta es cero, el resultado debe ser cero:

p(x,y)*ln(p(x,y)/(p(x)p(y)))		
0,05	0,14	0,00
-0,04	0,00	0,13
0,03	0,00	0,03

La MI corresponde a la suma de estos términos. En este caso la unidad es "nats" porque se utilizó el logaritmo natural. Para utilizar la base 2 y llevar el resultado a "bits" basta dividir por Ln(2):

MI en base e:	0,331
Ln(2)	0,693
MI en base 2:	0,477

Es decir, la variable X entrega 0,693 bits de información sobre la variable Y.

En este ejemplo, el analista de crédito podría calcular la MI entre otras variables (*features*) distintas de X para identificar las que tienen mayor poder predictivo sobre la variable de interés Y, en este caso los incumplimientos en las obligaciones de los créditos.

MI es una herramienta utilizada en *Machine Learning* y como es de esperar existen librerías de Python que realizan estos cálculos. La librería pública de Scikit incluye las funciones mutual_info_classif() para variables *target* discretas[2], y mutual_info_regression() para variables *target* continuas[3].

A continuación se presentan algunos ejemplos simplificados del uso de MI para identificar relaciones no lineales mediante códigos de Pyhton que utilizan esta librería. En todos ellos la variable X corresponde a los enteros de 1 a 100.

Junto a esta columna se publica también un archivo Excel con los datos y un código en Python con todos los ejemplos, incluyendo el de variables discretas.

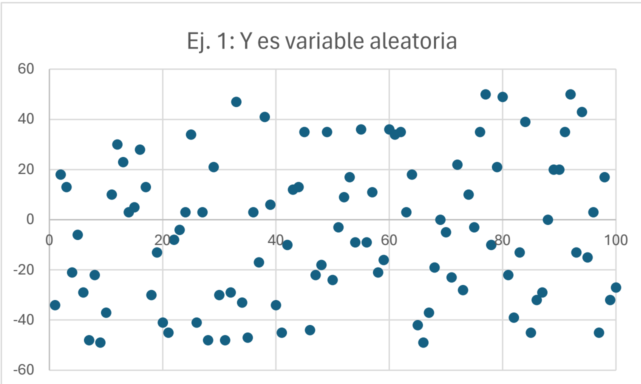
Para empezar, se analiza en caso en que la variable Y es aleatoria con distribución uniforme entre -50 y +50. Como es esperable, MI es cero ya que la variable X no entrega nada de información sobre la variable Y. El coeficiente de Pearson en también cercano a cero.

[2] https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_classif.html

[3] https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_regression.html

Ej1 MI: [0.]

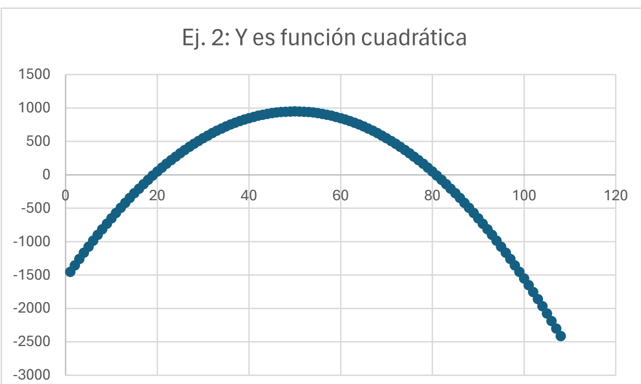
Pearson: 0.15021242553827382



En el segundo ejemplo, Y es una función cuadrática. En este caso MI logra identificar una relación entre las dos variables, a pesar de que no es lineal. Por su parte, y como es esperable, el coeficiente de Pearson es cercano a cero ya que sólo mide efectos lineales.

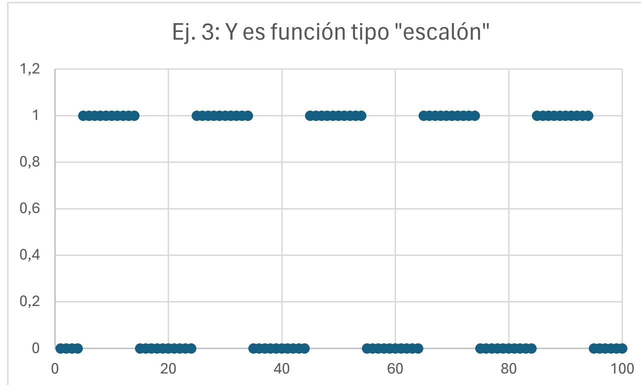
Ej2 MI: [2.51702455]

Pearson: -0.03870854961123829



En el tercer caso Y es una función “escalón” dada por:

=RESIDUO(REDONDEAR(X/10;0);2)



En este caso nuevamente MI es capaz de identificar una relación, mientras que el coeficiente de Pearson es muy cercano a 0 ya que la relación no es lineal:

Ej3 MI: [0.75910599]

Pearson: -0.034642748332099786

Así, el uso de MI en el análisis de relaciones entre variables puede ser una herramienta muy útil en el análisis de datos.

Esta columna es sólo una breve introducción para dar a conocer el concepto y a la aplicación de MI. El analista profesional interesado en aplicar esta herramienta debe estudiar el tema para conocer su aplicación y limitaciones, para lo que existe abundante información disponible en línea.

ANEXO: EJEMPLO EN PYTHON

```
# Este código está basado en el ejemplo presentado en:
#         https://scikit-learn.org/stable/auto_examples/feature_selection/plot_f_test_vs_mi.html#sphx-glr-auto-examples-feature-selection-plot-f-test-vs-mi-py

import xlwings as xw
import numpy as np
import math
from scipy.stats import pearsonr
from sklearn.feature_selection import mutual_info_regression, mutual_info_classif

# 1.- Variables discretas
ws = xw.Book("Cálculos mutual information.xlsx").sheets['Ej. 0']
data = ws.range("K16:L33").value
data_array = np.asarray(data)

Xbase = data_array[:,0]
X = Xbase.reshape(-1, 1)
y = data_array[:,1]

mi = mutual_info_classif(X, y, discrete_features=True) / math.log(2)
print("Ej. 0 MI: ", mi, "\n")

# 2.- Variables contínuas
ejemplos = ['Ej. 1', 'Ej. 2', 'Ej. 3']

for ej in ejemplos:
    ws = xw.Book("Cálculos mutual information.xlsx").sheets[ej]
    data = ws.range("B2:C101").value
    data_array = np.asarray(data)

    Xbase = data_array[:,0]
    X = Xbase.reshape(-1, 1)
    y = data_array[:,1]

    mi = mutual_info_regression(X, y) / math.log(2)
    print(ej, " MI: ", mi)

    X = list(Xbase)
    X = np.array(X)

    Pearson = pearsonr(X, y).statistic
    print("Pearson: ", Pearson, "\n")
```