Análisis de observaciones influyentes



Adrián Yared Armas de la Nuez

**Contenido**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

[**1. Actividad 1 3**](#_2c35568x6wg8)

[**1.1 Enunciado 3**](#_labmoglncig7)

[**1.2 Pregunta 1 3**](#_z5pepdq8gcss)

[**1.2.1 Apartado 1 3**](#_f6xix8il172b)

[**1.2.2 Resolución 3**](#_p8m57kfildsp)

[**1.2.2.1 Código 3**](#_b0t22d40xw0b)

[**1.2.2.2 Ejecución 3**](#_owfgyitvio5)

[**1.3.1 Apartado 2 3**](#_ng536bk0o14l)

[**1.3.2 Resolución 4**](#_i2eo7vtqe4ed)

[**1.3.2.1 Código 4**](#_63sst4aexsxc)

[**1.3.2.2 Ejecución 5**](#_ege9intxvs4)

[**1.4.1 Apartado 2 5**](#_mmmfsu45bv7y)

[**1.4.2 Resolución 5**](#_kpvogbdjnyuf)

[**1.4.2.1 Código 5**](#_f702m1nen3ai)

[**1.4.2.2 Ejecución 6**](#_5fizpewvbwrw)

[**2. Actividad 2 6**](#_ndd9o0gywiai)

[**2.1 Enunciado 6**](#_oevfml998z2m)

[**2.2 Pregunta 2 6**](#_9yx896p6w6g9)

[**2.2.1 Apartado 1 7**](#_4e368datkbnh)

[**2.2.2 Resolución 7**](#_sk2z3wg35asp)

[**2.2.2.1 Código 7**](#_r0edwcv5f32g)

[**2.2.2.2 Ejecución 7**](#_zfmet9or0790)

[**2.2.2.3 Explicación 8**](#_dju9xwmy2grx)

[**2.3.1 Apartado 2 8**](#_8ghg0w7xynze)

[**2.3.2 Resolución 8**](#_fb4hqjj4gdu4)

[**2.3.2.1 Código 8**](#_rp5fgg3koy9x)

[**2.3.2.2 Ejecución 8**](#_l1lbl5iua4t6)

[**2.3.2.3 Explicación 8**](#_nqzt31c1u3ze)

[**2.4.1 Apartado 3 9**](#_3r8v881jvy51)

[**2.4.2 Resolución 9**](#_owh6v7vknu77)

[**2.4.2.1 Código 9**](#_tooad918ti7)

[**2.4.2.2 Ejecución 10**](#_48na4aixsvfz)

[**2.4.2.3 Explicación 10**](#_1f85yjgc4x00)

[**2.5.1 Apartado 4 10**](#_fsscf47e1fym)

[**2.5.2 Resolución 10**](#_xigdudowgi6y)

[**2.5.2.1 Código 10**](#_2i9v5cf22izk)

[**2.5.2.2 Ejecución 11**](#_s3jvh029fs9k)

[**2.5.2.3 Explicación 11**](#_45g2tzcuoxd9)

[**2.6.1 Apartado 5 12**](#_2ebjes9r7oyh)

[**2.6.2 Resolución 12**](#_z8m3a75y5ozm)

[**2.6.2.1 Código 12**](#_amq14oemia8v)

[**2.6.2.2 Ejecución 12**](#_g05lvaqf4smt)

[**2.6.2.3 Explicación 12**](#_w3vm6jnfy1ky)

[**3. Colab y github 13**](#_ljfo1pgbg9nv)

## 

## 

## 

## 

## 

## 

## **1. Actividad 1**

### **1.1 Enunciado**

A partir del código de ejemplo utilizado en el notebook Ejemplo\_2\_4\_Observaciones\_influyentes\_Sin soluciones.ipynb Url:

<https://colab.research.google.com/drive/11JM5daNQUCB_VSAOHmpFjHmFuZDsB3-i?usp=sharing>

### **1.2 Pregunta 1**

#### **1.2.1 Apartado 1**

Calcular la media y la mediana antes de realizar la modificación de

incluir unos ingresos de 500.000€

#### **1.2.2 Resolución**

##### **1.2.2.1 Código**

# Calcular la media y mediana antes de incluir el ingreso de 500,000€

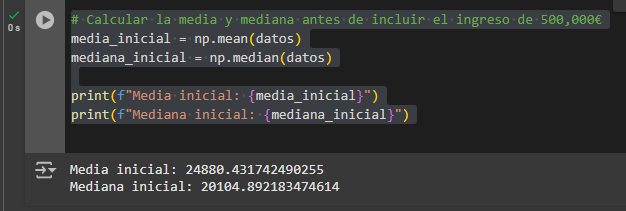
media\_inicial = np.mean(datos)

mediana\_inicial = np.median(datos)

print(f"Media inicial: {media\_inicial}")

print(f"Mediana inicial: {mediana\_inicial}")

##### **1.2.2.2 Ejecución**



#### **1.3.1 Apartado 2**

Aplicar el método de Probabilidad global, para detectar los outliers

utilizado en el ejemplo 2\_3\_Outliers

Url:

<https://colab.research.google.com/drive/1C6uBUxui_Qq9ee-51ycVYcqigrSHSZNY?usp=sharing>

#### **1.3.2 Resolución**

##### **1.3.2.1 Código**

import scipy.stats as st

# Probabilidad global

p\_g = 0.95

alfa\_g = (1 - p\_g) / 2

# Probabilidad ajustada para un dato

alfa = 1 - (1 - alfa\_g) \*\* (1 / len(datos))

Z\_alfa = st.norm.ppf(1 - alfa / 2)

# Intervalo de aceptación

xL = round(np.mean(datos) - Z\_alfa \* np.std(datos), 4)

xU = round(np.mean(datos) + Z\_alfa \* np.std(datos), 4)

# Identificación de outliers

outliers = [i for i, val in enumerate(datos) if val < xL or val > xU]

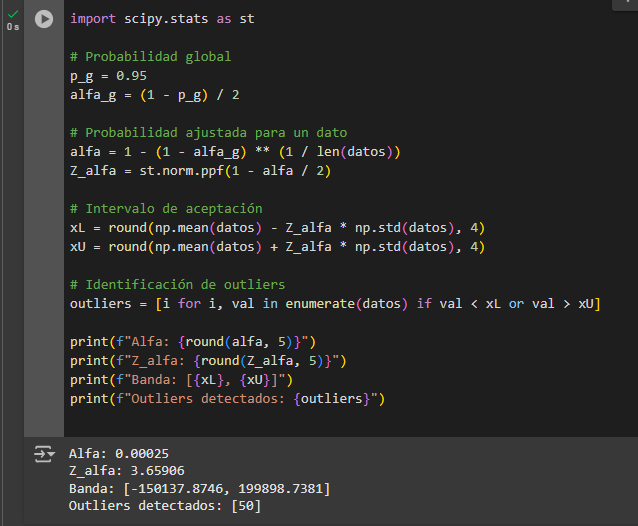
print(f"Alfa: {round(alfa, 5)}")

print(f"Z\_alfa: {round(Z\_alfa, 5)}")

print(f"Banda: [{xL}, {xU}]")

print(f"Outliers detectados: {outliers}")

##### **1.3.2.2 Ejecución**



#### **1.4.1 Apartado 2**

Repetir el mismo procedimiento de detectar los outliers para la

mediana: ¿Qué ocurre?

#### **1.4.2 Resolución**

##### **1.4.2.1 Código**

# Detectar outliers usando el rango intercuartil (IQR) basado en la mediana

q1 = np.percentile(datos, 25)

q3 = np.percentile(datos, 75)

iqr = q3 - q1

# Límites inferior y superior usando el rango intercuartil

limite\_inferior = q1 - 1.5 \* iqr

limite\_superior = q3 + 1.5 \* iqr

outliers\_mediana = [i for i, val in enumerate(datos) if val < limite\_inferior or val > limite\_superior]

print(f"Cuartil 1: {q1}")

print(f"Cuartil 3: {q3}")

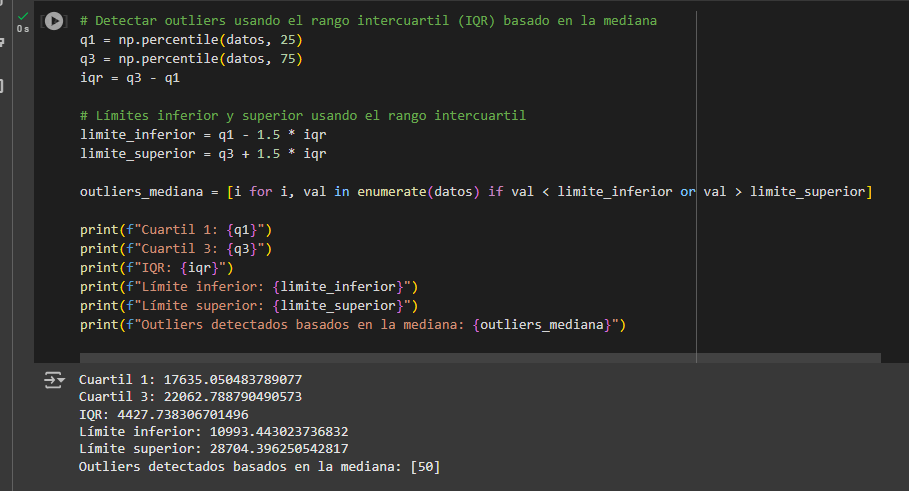
print(f"IQR: {iqr}")

print(f"Límite inferior: {limite\_inferior}")

print(f"Límite superior: {limite\_superior}")

print(f"Outliers detectados basados en la mediana: {outliers\_mediana}")

##### **1.4.2.2 Ejecución**



## **2. Actividad 2**

### **2.1 Enunciado**

A partir del código de ejemplo utilizado en el notebook

Ejemplo\_2\_5\_Escalamiento\_de\_datos\_Sin soluciones.ipynb

Url: <https://colab.research.google.com/drive/11vLMbjw5XmF7dJks0b04gfMdCnClE0Kw?usp=sharing>

### **2.2 Pregunta 2**

Considerar que la variable X toma los valores 1,2,3,4,5,6,7,8,9,10. Se pide:

#### **2.2.1 Apartado 1**

¿Cuánto vale la media, mediana, la desviación estándar muestral, la varianza muestral y el rango de la variable X?

#### **2.2.2 Resolución**

##### **2.2.2.1 Código**

x = np.array ([1,2,3,4,5,6,7,8,9,10])

media = np.mean(x)

mediana = np.median(x)

desviacion\_estandar = np.std(x)

varianza = np.var(x)

rango = np.ptp(x)

print(f"Media: {media}")

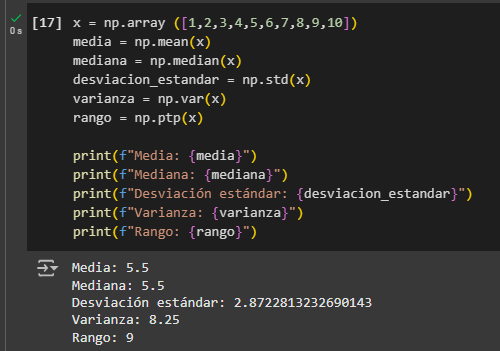
print(f"Mediana: {mediana}")

print(f"Desviación estándar: {desviacion\_estandar}")

print(f"Varianza: {varianza}")

print(f"Rango: {rango}")

##### **2.2.2.2 Ejecución**



##### **2.2.2.3 Explicación**

El código calcula estadísticas descriptivas de un arreglo x que contiene los números del 1 al 10. La media y la mediana son ambas 5.5, reflejando la simetría de los datos. La desviación estándar es aproximadamente 2.87, indicando la dispersión promedio de los valores respecto a la media, mientras que la varianza (8.25) mide la dispersión al cuadrado. Finalmente, el rango (9) representa la diferencia entre el valor máximo (10) y el mínimo (1), mostrando la amplitud total de los datos.

#### **2.3.1 Apartado 2**

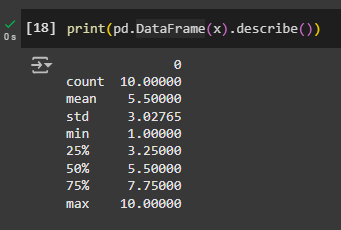
Utilizar la función describe() de Panda, para obtener la media, desviación estándar, etc…

#### **2.3.2 Resolución**

##### **2.3.2.1 Código**

print(pd.DataFrame(x).describe())

##### **2.3.2.2 Ejecución**



##### **2.3.2.3 Explicación**

El código pd.DataFrame(x).describe() genera un resumen estadístico de la variable x, que contiene 10 valores del 1 al 10. Muestra que hay 10 datos (count), con una media y mediana de 5.5 (mean y 50%), una desviación estándar de 3.02765 (std), un rango que va desde el mínimo 1.0 (min) al máximo 10.0 (max), y percentiles clave como el 25% (3.25) y 75% (7.75). Esto resume de forma compacta la distribución y dispersión de los datos.

#### **2.4.1 Apartado 3**

¿Por qué el resultado de calcular la desviación estándar con Numpy es diferente a la calculada por describe de Panda? ¿Qué ajuste sería necesario realizar para que los resultados fuesen similares/iguales?

#### **2.4.2 Resolución**

##### **2.4.2.1 Código**

import numpy as np

import pandas as pd

# Datos de ejemplo

data = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# Numpy: desviación estándar poblacional

std\_numpy\_pop = np.std(data) # Por defecto ddof=0

print("Numpy (poblacional):", std\_numpy\_pop)

# Numpy: desviación estándar muestral

std\_numpy\_sample = np.std(data, ddof=1) # Igual que describe() de Pandas

print("Numpy (muestral):", std\_numpy\_sample)

# Pandas: desviación estándar (muestral)

std\_pandas = pd.Series(data).std() # Por defecto ddof=1

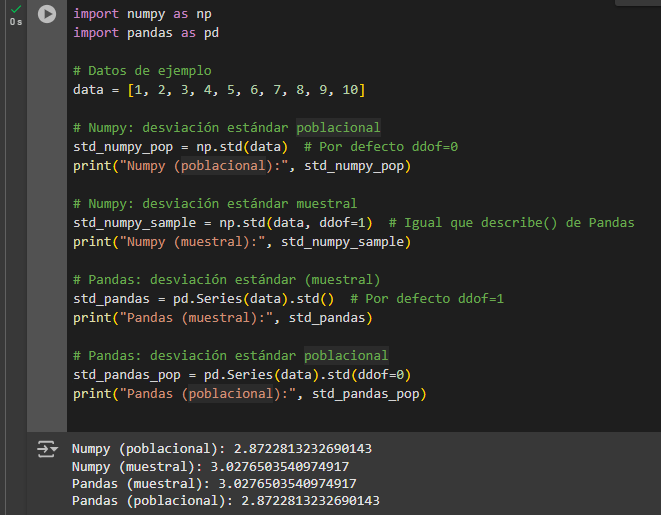
print("Pandas (muestral):", std\_pandas)

# Pandas: desviación estándar poblacional

std\_pandas\_pop = pd.Series(data).std(ddof=0)

print("Pandas (poblacional):", std\_pandas\_pop)

##### **2.4.2.2 Ejecución**



##### **2.4.2.3 Explicación**

La diferencia surge porque Numpy, por defecto, calcula la desviación estándar poblacional (dividiendo por 𝑁), mientras que Pandas' describe usa la desviación estándar muestral (dividiendo por 𝑁−1, que es el sesgo corregido o "Bessel's correction"). Para hacer que los resultados sean iguales, puedes ajustar Numpy para que use el mismo método muestral estableciendo ddof=1 en la función np.std. Así, ambos cálculos usarán la corrección 𝑁−1.

#### **2.5.1 Apartado 4**

Estandarizar la variable (escalamiento) mediante rangos y a continuación calcular la media y la mediana de la variable escalada.

#### **2.5.2 Resolución**

##### **2.5.2.1 Código**

import numpy as np

import pandas as pd

# Datos

X = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])

# Escalamiento mediante rangos: (X - min(X)) / (max(X) - min(X))

X\_scaled = (X - X.min()) / (X.max() - X.min())

# Cálculo de la media y mediana de la variable escalada

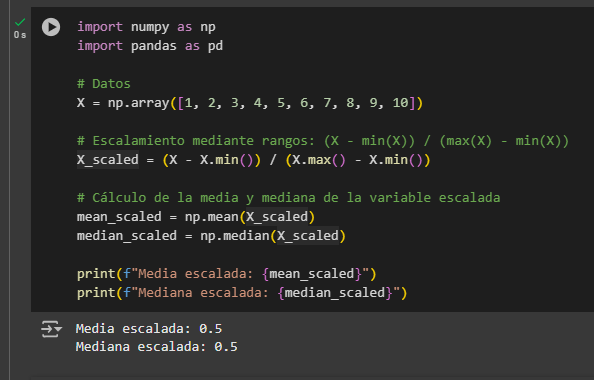
mean\_scaled = np.mean(X\_scaled)

median\_scaled = np.median(X\_scaled)

print(f"Media escalada: {mean\_scaled}")

print(f"Mediana escalada: {median\_scaled}")

##### **2.5.2.2 Ejecución**



##### **2.5.2.3 Explicación**

El escalamiento mediante rangos transforma los valores de la variable X al intervalo [0, 1], manteniendo las proporciones relativas. En este caso, la distribución original de X es uniforme y simétrica, por lo que su valor central (mediana) y promedio (media) también están en el punto medio del rango escalado, es decir, 0.5. Este resultado refleja que la transformación no altera la simetría de la distribución. Además, la media y mediana coinciden porque la distribución permanece uniforme tras el escalado.

#### **2.6.1 Apartado 5**

Repetir el apartado anterior con el escalamiento Z - score

#### **2.6.2 Resolución**

##### **2.6.2.1 Código**

# Escalamiento Z-score: (X - mean(X)) / std(X)

X\_zscore = (X - X.mean()) / X.std()

# Cálculo de la media y mediana de la variable escalada

mean\_zscore = np.mean(X\_zscore)

median\_zscore = np.median(X\_zscore)

print(f"Media escalada z-score: {mean\_zscore}")

print(f"Mediana escalada z-score: {median\_zscore}")

##### **2.6.2.2 Ejecución**

### 

##### **2.6.2.3 Explicación**

El escalamiento Z-score transforma los datos para que tengan una media de 0 y una desviación estándar de 1. En este caso, la media calculada es aproximadamente 0 (con un error numérico insignificante, ~10^−17, lo cual confirma el centrado. La mediana de 0 indica que el valor central de los datos escalados también coincide con el centro de la distribución normalizada. Este método es útil para comparar datos en diferentes escalas, ya que elimina efectos de magnitud o unidad original.

## **3. Colab y github**

[](https://colab.research.google.com/drive/1ePJQs9d-IkHQsX10ceuapA8dS6oDFSUj?usp=sharing)