

## Análisis de observaciones influyentes



Adrián Yared Armas de la Nuez



## Contenido

---

<b>1. Actividad 1.....</b>	<b>3</b>
<b>1.1 Enunciado.....</b>	<b>3</b>
<b>1.2 Pregunta 1.....</b>	<b>3</b>
<b>1.2.1 Apartado 1.....</b>	<b>3</b>
<b>1.2.2 Resolución.....</b>	<b>3</b>
<b>1.2.2.1 Código.....</b>	<b>3</b>
<b>1.2.2.2 Ejecución.....</b>	<b>3</b>
<b>1.3.1 Apartado 2.....</b>	<b>3</b>
<b>1.3.2 Resolución.....</b>	<b>4</b>
<b>1.3.2.1 Código.....</b>	<b>4</b>
<b>1.3.2.2 Ejecución.....</b>	<b>5</b>
<b>1.4.1 Apartado 2.....</b>	<b>5</b>
<b>1.4.2 Resolución.....</b>	<b>5</b>
<b>1.4.2.1 Código.....</b>	<b>5</b>
<b>1.4.2.2 Ejecución.....</b>	<b>6</b>
<b>2. Actividad 2.....</b>	<b>6</b>
<b>2.1 Enunciado.....</b>	<b>6</b>
<b>2.2 Pregunta 2.....</b>	<b>6</b>
<b>2.2.1 Apartado 1.....</b>	<b>7</b>
<b>2.2.2 Resolución.....</b>	<b>7</b>
<b>2.2.2.1 Código.....</b>	<b>7</b>
<b>2.2.2.2 Ejecución.....</b>	<b>7</b>
<b>2.2.2.3 Explicación.....</b>	<b>8</b>
<b>2.3.1 Apartado 2.....</b>	<b>8</b>
<b>2.3.2 Resolución.....</b>	<b>8</b>
<b>2.3.2.1 Código.....</b>	<b>8</b>
<b>2.3.2.2 Ejecución.....</b>	<b>8</b>
<b>2.3.2.3 Explicación.....</b>	<b>8</b>
<b>2.4.1 Apartado 3.....</b>	<b>9</b>
<b>2.4.2 Resolución.....</b>	<b>9</b>
<b>2.4.2.1 Código.....</b>	<b>9</b>
<b>2.4.2.2 Ejecución.....</b>	<b>10</b>
<b>2.4.2.3 Explicación.....</b>	<b>10</b>
<b>2.5.1 Apartado 4.....</b>	<b>10</b>
<b>2.5.2 Resolución.....</b>	<b>10</b>
<b>2.5.2.1 Código.....</b>	<b>10</b>



## Análisis de observaciones influyentes

2.5.2.2 Ejecución.....	11
2.5.2.3 Explicación.....	11
2.6.1 Apartado 5.....	12
2.6.2 Resolución.....	12
2.6.2.1 Código.....	12
2.6.2.2 Ejecución.....	12
2.6.2.3 Explicación.....	12
3. Colab y github.....	13

# 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](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€

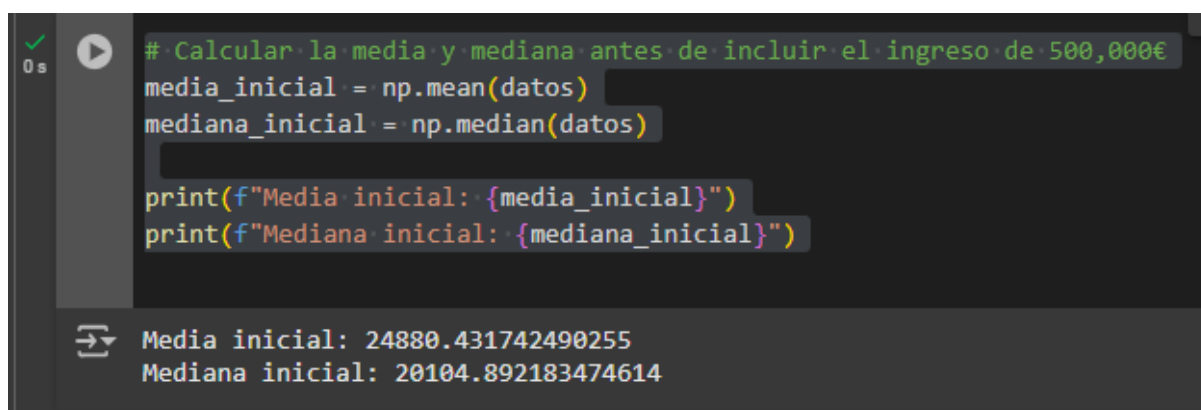
### 1.2.2 Resolución

#### 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



```
# 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}")
```

Media inicial: 24880.431742490255  
Mediana inicial: 20104.892183474614

### 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](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

```
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}")
```

Alfa: 0.00025  
Z\_alfa: 3.65906  
Banda: [-150137.8746, 199898.7381]  
Outliers detectados: [50]

### 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

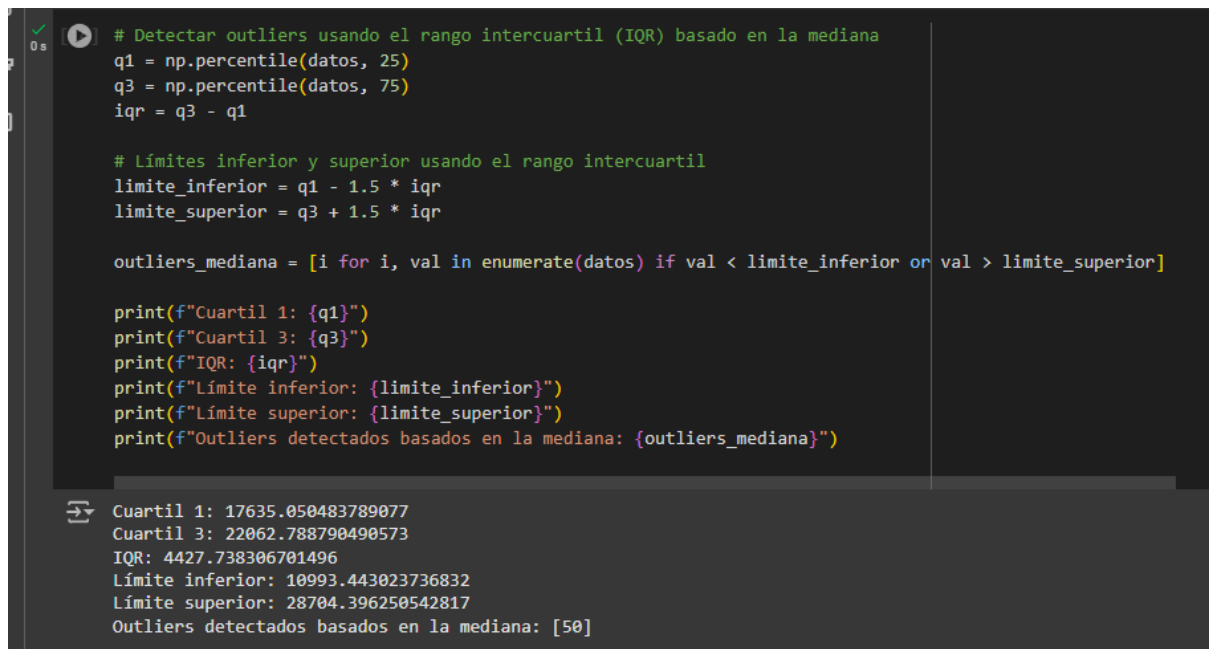
```
# Detectar outliers usando el rango intercuartil (IQR) basado en la mediana
q1 = np.percentile(datos, 25)
q3 = np.percentile(datos, 75)
iqr = q3 - q1

# 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



```
# 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}")
```

Cuartil 1: 17635.050483789077  
Cuartil 3: 22062.788790490573  
IQR: 4427.738306701496  
Límite inferior: 10993.443023736832  
Límite superior: 28704.396250542817  
Outliers detectados basados en la mediana: [50]

## 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

```
✓ [17] x = np.array ([1,2,3,4,5,6,7,8,9,10])
0s 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}")
```

⇒ Media: 5.5  
Mediana: 5.5  
Desviación estándar: 2.8722813232690143  
Varianza: 8.25  
Rango: 9



### 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

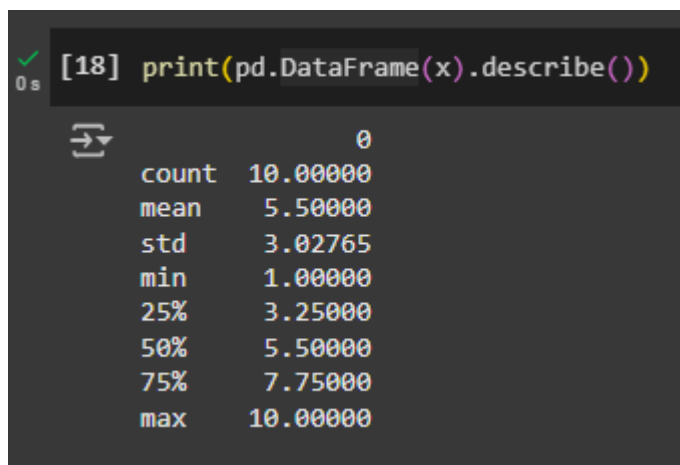
Utilizar la función `describe()` de Panda, para obtener la media, desviación estándar, etc...

### 2.3.2 Resolución

#### 2.3.2.1 Código

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

#### 2.3.2.2 Ejecución



	0
count	10.00000
mean	5.50000
std	3.02765
min	1.00000
25%	3.25000
50%	5.50000
75%	7.75000
max	10.00000

#### 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

```
✓ 0 s ▶ 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)
```

```
↳ Numpy (poblacional): 2.8722813232690143
Numpy (muestral): 3.0276503540974917
Pandas (muestral): 3.0276503540974917
Pandas (poblacional): 2.8722813232690143
```

### 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

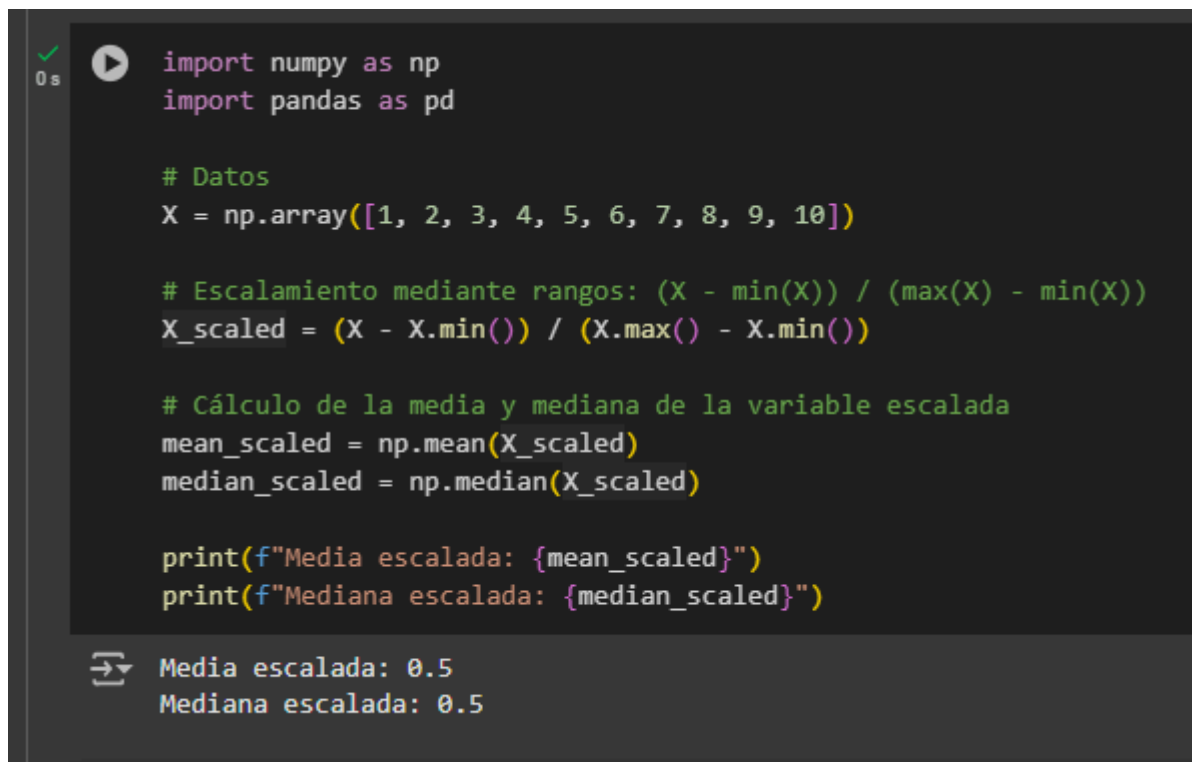
# Datos
X = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])

# 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



```
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}")
```

Media escalada: 0.5  
Mediana escalada: 0.5

### 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

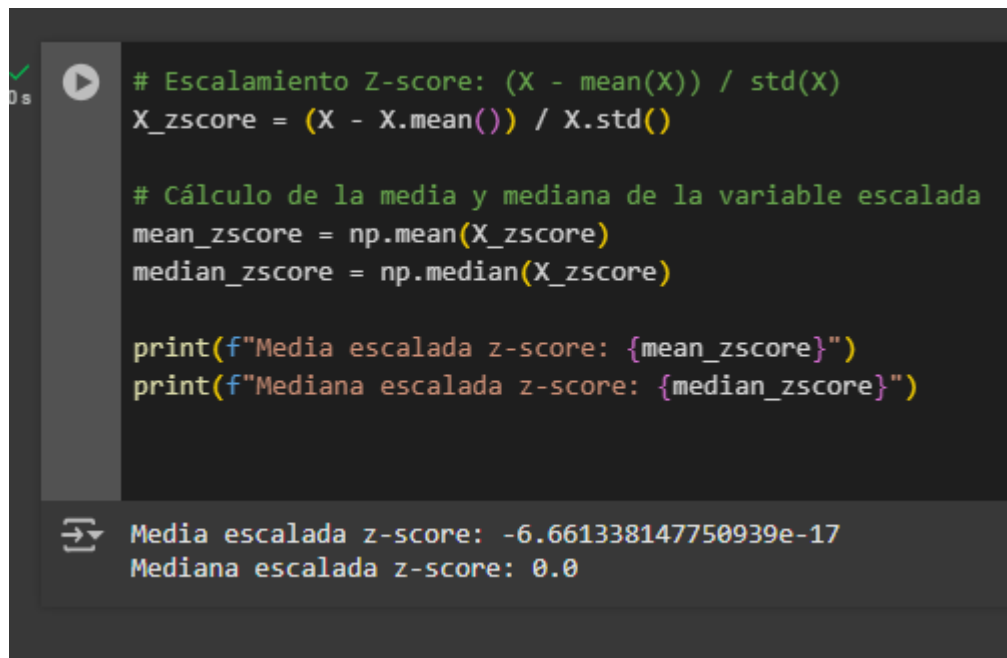
#### 2.6.2.1 Código

```
# 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



```
# 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}")
```

Media escalada z-score: -6.661338147750939e-17  
Mediana escalada z-score: 0.0

#### 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,  $\sim 10^{-17}$ , 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

