



Noviembre 2023

# ACT2\_1: ANÁLISIS DE OBSERVACIONES INFLUYENTES



Carolina María Montesdeoca Álvarez  
SNS

1. A partir del código de ejemplo utilizado en el notebook

Ejemplo\_2\_4\_Observaciones\_influyentes\_Sin soluciones.ipynb

Responder a las siguientes preguntas:

- a) Calcular la media y la mediana antes de realizar la modificación de incluir unos ingresos de 500.000€.

**Ejercicio:** Calcular la media y la mediana antes de realizar la modificación de incluir unos ingresos de 500.000€

```
In [2]: media= np.mean(datos)
        mediana= np.median (datos) # Es el valor central cuando los datos se ordenan
        print(f" Media= {media} y Mediana={mediana}")

Media= 20096.069596407473 y Mediana=19616.96170362
```

```
In [3]: # El vecino 50 tiene unos ingresos significativamente mayores que el resto
        datos[50]=500000
        datos
```

```
Out[3]: array([ 15088.16866976,  15418.19840141,  15734.54993286,  15863.1891019 ,
                15900.99199346,  15976.19091218,  16065.28740846,  16076.0803968 ,
                16080.396604 ,  16093.87394362,  16230.99569423,  16356.30883638,
                16391.05471534,  16399.53666727,  16429.76520489,  16430.22605648,
                16472.23814912,  16506.25601234,  16608.00857256,  16680.78643896,
                17136.82562646,  17257.38914481,  17449.33587048,  17473.22462941,
                17787.49153553,  17796.59857265,  17863.99189135,  17866.57170504,
                17930.00086218,  18029.23605381,  18151.20251057,  18178.66144402,
                18247.28942281,  18349.6148917 ,  18359.73400586,  18368.76631486,
                18399.20440442,  18447.24845996,  18491.64362995,  18542.39963897,
                18629.93657956,  18659.447816 ,  18707.35776162,  18782.06963533,
                19120.55590623,  19300.44547943,  19390.06586292,  19490.66140063,
                19576.41846511,  19592.23525297,  500000. ,  19703.09323869,
                20131.79000655,  20144.40869092,  20259.97884259,  20363.92036153,
                20765.0093 ,  20921.01678918,  21041.00705409,  21166.05660042,
                21238.50433427,  21248.98951606,  21277.57019136,  21386.00467423,
                21700.45079257,  21828.44569896,  21918.36186635,  22109.41467321,
                22277.62252983,  22375.69143584,  22648.32723602,  22697.16037225,
                22719.95125574,  22854.96742846,  22975.00823177,  23036.68135264,
                23049.10493756,  23154.25832262,  23291.69006201,  23341.4119319 ,
                23397.56504815,  23592.19904231,  23596.36807568,  23736.8059911 ,
                23753.90626155,  23911.15313045,  24060.39549931,  24093.15899423,
                24178.81818209,  24184.99823939,  24194.45002281,  24253.16923715,
                24402.71819937,  24422.58763433,  24497.08766578,  24703.49693113,
                24720.73467051,  24783.36642087,  24803.15924836,  24875.50677766])
```

Cálculo de media y mediana en toda la muestra

```
In [4]: media= np.mean(datos)
        mediana= np.median (datos) # Es el valor central cuando los datos se ordenan
        print(f" Media= {media} y Mediana={mediana}")

Media= 24899.652714864784 y Mediana=19647.664245831245
```

Notar que los ingresos del barrio son aprox. 20000 euros. El hecho de que se haya trasladado un vecino rico ha incrementado aprox. en 5000 euros la media. La mediana es insensible a este hecho. Se trata de identificar que datos son influyentes para los dos procedimientos estadísticos

- b) Aplicar el método de Probabilidad global, para detectar los outliers utilizado en el ejemplo 2\_3\_Outliers.

Aplicando el método describe()

**Ejercicio:** Aplicar el método de describe para ver un resumen estadístico de los datos.

```
In [5]: pd.DataFrame(datos).describe()
```

```
Out[5]:
```

	0
count	100.000000
mean	24899.652715
std	48081.947522
min	15088.168670
25%	17794.321813
50%	19647.664246
75%	23039.787249
max	500000.000000

Aplicando el método de Probabilidad Global:

**Ejercicio:** Aplicar el método de Probabilidad global, para detectar los outliers utilizado en el ejemplo 2\_3\_Outliers

```
In [6]: # CRITERIO 1: PROBABILIDAD GLOBAL - Explicado en el capítulo 5 - Preprocesamiento (Semana 2), en la pág 10
# Asumiendo que las variables tienen una distribución normal.
# Probabilidad de la muestra de estar dentro de las bandas
p_g=0.95
# probabilidad global
alfa_g=(1-p_g)/2
# probabilidad para un solo dato
alfa = 1-(1-alfa_g)**(1/len(datos)) # Se realiza este ajuste para ser más precisos.
```

```
In [7]: import scipy.stats as st
## CRITERIO 2: Criterio Chauvenet
# alfa=1/(2*len(datos))
Z_alfa=st.norm.ppf(1-alfa/2)
# Impresión de resultados
alfa=round(alfa,5)
Z_alfa=round(Z_alfa,5)
print(f" Alfa = {alfa}")
print(f" Z_alfa = {Z_alfa}")
```

```
Alfa =0.00025
Z_alfa =3.65906
```

Cálculo de bandas

Truco: Si los datos tienen una distribución normal calcula estas bandas con la fórmula/técnica propuesta en las siguientes celdas

```
In [8]: xL= round(np.mean(datos)-Z_alfa* np.std(datos),4)
xU= round(np.mean(datos)+Z_alfa* np.std(datos),4)
print(f" Banda= [ {xL},{xU}]")
```

```
Banda= [ -150153.1943,199952.4997]
```

```
In [9]: for i in range(len(datos)):
        if datos[i] < xL or datos[i]>xU:
            print(f" El dato[{i}]= {datos[i]} es un outlier")
```

```
El dato[50]=500000.0 es un outlier
```

c) Repetir el mismo procedimiento de detectar los outliers para la mediana:  
¿Qué ocurre?

**Ejercicio:** Repetir el mismo procedimiento de detectar los outliers para la mediana: ¿Qué ocurre?

```
In [13]: xL= round(np.median(datos)-Z_alfa* np.std(datos),4)
xU= round(np.median(datos)+Z_alfa* np.std(datos),4)
print(f" Banda= [ {xL},{xU}]")

Banda= [ -155405.1828,194700.5112]

In [14]: for i in range(len(datos)):
if datos[i] < xL or datos[i]>xU:
print(f" El dato[{i}]= {datos[i]} es un outlier")

El dato[50]=500000.0 es un outlier
```

En la mayoría de los casos, los outliers tienen influencia en la media, pero no en la mediana, o la moda. Por lo tanto, los outliers son importantes en su efecto en la media. La mediana es insensible a este hecho.

2. A partir del código de ejemplo utilizado en el notebook

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

Responder a las siguientes preguntas:

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

a) (2 Puntos) ¿Cuánto vale la media, mediana, la desviación estándar muestral, la varianza muestral y el rango de la variable X?

Considerar que la variable X toma los valores 1,2,3,4,5,6,7,8,9,10. Se pide: ¿Cuánto vale la media, mediana, la desviación estándar muestral, la varianza muestral y el rango de la variable X?

```
In [8]: import numpy as np
X=[1,2,3,4,5,6,7,8,9,10]
print("La media es:", np.mean(X))
print("La mediana es:", np.median(X))
print("La desviación estándar muestral es:", np.std(X))
print("La varianza muestral es:", np.var(X))
print("El rango de la variable X es:", np.ptp(X,0))

La media es: 5.5
La mediana es: 5.5
La desviación estándar muestral es: 2.8722813232690143
La varianza muestral es: 8.25
El rango de la variable X es: 9
```

- b) (1 Punto) Utilizar la función describe() de Panda, para obtener la media, desviación estándar, etc...

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

```
In [65]: X = pd.DataFrame(X, columns=['X'])
X.describe()
```

Out[65]:

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

- c) (1 Punto) ¿Por qué el resultado de calcular la desviación estándar con Numpy es diferente a la calculada por describe de Panda?

La diferencia se debe a que Pandas por defecto usa un grado de libertad al calcular la varianza, usando como denominador  $n - 1$ , mientras que NumPy por defecto usa 0 grados de libertad, usando como denominador  $n$  directamente (donde  $n$  es el tamaño de la muestra). Esto está determinado por el argumento **ddof** y es lo que se conoce como *corrección de Bessel*.

Es decir, la fórmula para calcular la varianza por defecto en NumPy sería:

$$s_n^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2$$

mientras que Pandas por defecto usa:

$$s^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$$

Esto es importante cuando se quiere estimar la desviación estándar como indicador estadístico de una población a partir de una muestra de la misma. NumPy por tanto proporciona por defecto una estimación sesgada de la varianza poblacional (sin la corrección de Bessel) mientras que Pandas proporciona una estimación no sesgada de la varianza de una población infinita hipotética (con la corrección de Bessel).

¿Qué ajuste sería necesario realizar para que los resultados fuesen similares/iguales?

Para obtener el mismo resultado en Pandas y en NumPy se modifica el parámetro

$ddof = 1 \rightarrow$  Corrección de Bessel (estimación no sesgada)

**Fuente:** Funciones estadísticas.

<https://es.stackoverflow.com/questions/279352/funciones-estad%C3%ADsticas-misma-funci%C3%B3n-resultados-diferentes-en-pandas-scipy-y-p>

d) (1 Punto) Estandarizar la variable (escalamiento) mediante rangos y a continuación calcular la media y la mediana de la variable escalada.

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

```
In [66]: # MinMaxScaler - Definimos el tipo de escalamiento a realizar: Estandarización por rangos
preprocessor_RR = ColumnTransformer(
    [['scale', MinMaxScaler(), ['X']], remainder='passthrough'])

Data_prep_RR = preprocessor_RR.fit_transform(X)

# Cálculo de la media y la mediana antes de introducir un valor atípico
print("Media =", np.mean(Data_prep_RR))
print("Mediana =", np.median(Data_prep_RR))

Media = 0.5
Mediana = 0.5
```

e) (1 Punto) Repetir el apartado anterior con el escalamiento Z - score

Repetir el apartado anterior con el escalamiento Z - score

```
In [67]: # Standard scaler - Definimos el tipo de escalamiento a realizar: estandarización Z-score
preprocessor_SS = ColumnTransformer(
    [ ('scale', StandardScaler(), ['X']) ], remainder='passthrough')

Data_prep_SS = preprocessor_SS.fit_transform(X)

# Cálculo de la media y la mediana antes de introducir un valor atípico
print("Media =", np.mean(Data_prep_SS))
print("Mediana =", np.median(Data_prep_SS))

Media = -6.661338147750939e-17
Mediana = 0.0
```

Indicar la URL del GitHub donde se encuentran el/los diferentes cuadernos que has utilizado, con el objeto de consultarlos para descargarlos y verificar su funcionamiento.

[https://github.com/carolProg/SNS\\_23\\_24](https://github.com/carolProg/SNS_23_24)