Análisis de observaciones influyentes

Javier Díaz Machado

1. 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

   Responder a las siguientes preguntas:

   a) **(1 Punto)** 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€

```
#Antes de insertar el dato en la posición 50, calculamos la media y mediana
media=  np.mean(datos)
mediana=  np.median (datos)
print(f" Media= {media} y Mediana={mediana}")
```

✓ 0.0s

```
Media= 24975.187573254763 y Mediana=20310.00047943289
```

```
# Simulamos (porque se lo asignamos en la siguiente línea) que el vecino 50 tiene unos ingresos significativamente mayores que el resto
datos[50]=500000
datos
```

   b) **(2 Puntos)** 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)

Ejercicio: Aplicar el método de Probabilidad global, para detectar los outliers utilizado en el ejemplo 2_3_Outliers

```
# Insertar el código aquí
# Probabilidad de la muestra de estar dentro de las bandas
p_g=0.90
alfa_g=(1-p_g)/2
print("Probabilidad global: ",alfa_g)
alfa= 1-(1-alfa_g)**(1/len(datos))
print("Probabilidad de un solo dato: ",alfa) #
```

✓ 0.0s

```
Probabilidad global:  0.04999999999999999
Probabilidad de un solo dato:  0.0005128014162623096
```

c) **(1 Punto)** Repetir el mismo procedimiento de detectar los outliers para la mediana: ¿Qué ocurre?

```
phi=np.zeros(len(datos))
for i in  range(len(datos)):
  datos_aux=datos
  datos_sin_i=np.delete(datos_aux, i)
  phi[i]=np.median(datos_sin_i)
# Aplicamos el método de la distancia entre cuartiles, en este caso, al aplicar el método Jackknife no obtenemos
# el valor del elemento que es considerado Outlier, sino la posición del mismo.
Q1 = np.quantile(phi,0.25)
Q3 = np.quantile(phi,0.75)
IQR = Q3 - Q1
xL=Q1 - 1.5 * IQR
xU=Q3 + 1.5 * IQR
OutliersN=0
for i in  range(len(datos)):

        if phi[i] < xL  or phi[i]>xU:
          OutliersN+=1
          print(f" El dato {i} es una observación influyente para la media")
if OutliersN==0:
  print("No hay observaciones influyentes, porque se ha utilizado la mediana como medida de dispersión")
```

✓ 0.0s

```
No hay observaciones influyentes, porque se ha utilizado la mediana como medida de dispersión
```

2. 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

   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?
   Código utilizado:

```
X = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
df = pd.DataFrame(X)
# Cálculos
media = np.mean(X)
mediana = np.median(X)
varianza_muestral = np.var(X, ddof=1)  # ddof=1 para la varianza muestral
desviacion_estandar_muestral = np.std(X, ddof=1)  # ddof=1 para la desviación estándar muestral
rango = np.ptp(X)  # Diferencia entre el máximo y el mínimo
print(f"Media: {media}")
print(f"Mediana: {mediana}")
print(f"Varianza muestral: {varianza_muestral}")
print(f"Desviación estándar muestral: {desviacion_estandar_muestral}")
print(f"Rango: {rango}")
```

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

Media: 5.5
Mediana: 5.5
Varianza muestral: 9.166666666666666
Desviación estándar muestral: 3.0276503540974917
Rango: 9
```

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

```
#b) (1 Punto) Utilizar la función describe() de Panda, para obtener la media, desviación estándar, etc...
print("\n  b) (1 Punto) Utilizar la función describe() de Panda, para obtener la media, desviación estándar, etc...")
print("resultado de df.describe():\n",df.describe())  #Lo imprimo porque de otra forma no se ve el resultado
```

```
  b) (1 Punto) Utilizar la función describe() de Panda, para obtener la media, desviación estándar, etc...
resultado de df.describe():
              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
```

c) **(1 Punto)** ¿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?

La "desviación estándar de la muestra" se calcula usando N-1 en el denominador, lo que se llama "corrección de Bessel". Esto corrige el sesgo hacia valores más bajos cuando se usa la media de la muestra en lugar de la media real de la población. Aunque la estimación sigue estando sesgada, es menos sesgada que sin la corrección. Para aplicar esta corrección en pandas, se usa ddof=1.

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

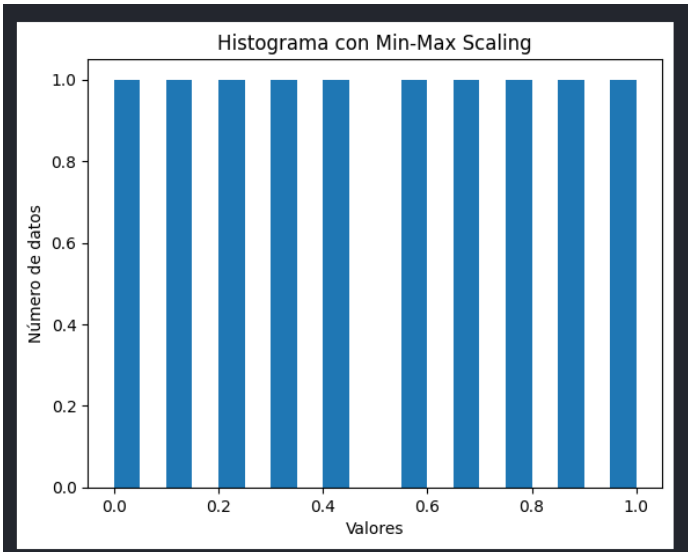
```
dfX = pd.DataFrame(X)

# Escalamiento mediante rangos
X_min = dfX.min()
X_max = dfX.max()
X_escalado = (dfX - X_min) / (X_max - X_min)

# Crear el histograma
plt.hist(X_escalado, bins=20)
plt.ylabel('Número de datos')
plt.xlabel('Valores')
plt.title('Histograma con Min-Max Scaling')
plt.show()

# Calcular la media y la mediana de la variable escalada
media = np.mean(X_escalado)
mediana = np.median(X_escalado)

print(f'Media de la variable escalada: {media}')
print(f'Mediana de la variable escalada: {mediana}')
```




```
Media de la variable escalada: 0.5
Mediana de la variable escalada: 0.5
```

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

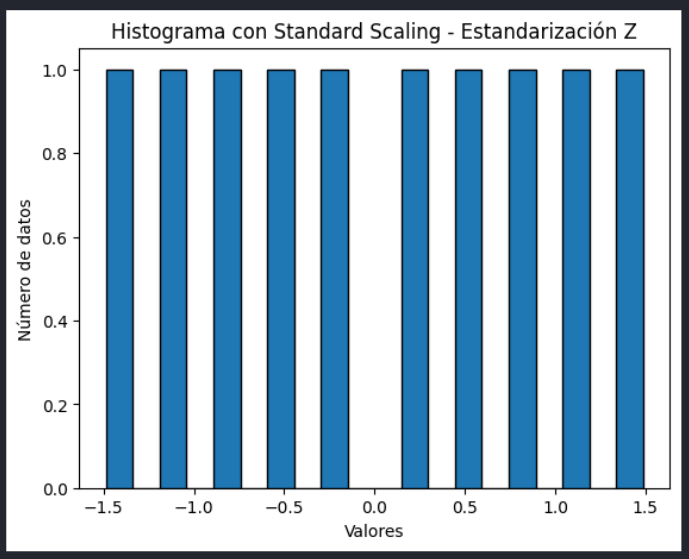
```python
# Estandarización Z
X_mean = dfX.mean()
X_std = dfX.std()
X_escalado = (dfX - X_mean) / X_std

# Crear el histograma
plt.hist(X_escalado, bins=20, edgecolor='black')
plt.ylabel('Número de datos')
plt.xlabel('Valores')
plt.title('Histograma con Standard Scaling - Estandarización Z')
plt.show()

# Calcular la media y la mediana de la variable escalada
media = np.mean(X_escalado)
mediana = np.median(X_escalado)

print(f'Media de la variable escalada: {media}')
print(f'Mediana de la variable escalada: {mediana}')
```

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




```
Media de la variable escalada: 4.4408920985006264e-17
Mediana de la variable escalada: 0.0
```