



SESIÓN CONCEPTOS BÁSICOS DE ESTADÍSTICA DESCRIPTIVA

CONTENIDOS:

- Tipos de variable (categóricas, cuantitativas, numéricas, discretas).
- Tablas de frecuencia.
- Medidas de tendencia central: media, mediana y moda.
- Medidas de dispersión: rango de variación, varianza y desviación estándar.
- población, muestra y corrección de bessel.
- Medidas de posición (cuartil, quintil, decil, percentil).
- Puntos atípicos.
- Gráficos y diagramas: histograma y boxplot.

TIPOS DE VARIABLE (CATEGÓRICAS, CUANTITATIVAS, NUMÉRICAS, DISCRETAS).

En el análisis de datos, es fundamental identificar el tipo de variable con la que estamos trabajando, ya que esto determina cómo podemos analizar, visualizar y manipular los datos. Las variables se pueden clasificar en diferentes tipos según su naturaleza y las características que representan. En este contexto, vamos a explorar los tipos de variables más comunes que encontramos en conjuntos de datos: categóricas, cuantitativas, numéricas y discretas.

Variables Categóricas

Representan atributos o características de un grupo sin un valor numérico asociado. No tienen un orden cuantificable.

Ejemplos:

- Género: Masculino, Femenino, Otro.
- Color favorito: Rojo, Azul, Verde.
- Tipo de transporte: Auto, Bicicleta, Autobús.



Tipos de Variables Categóricas:

1. Nominales: No tienen un orden específico (Ejemplo: colores).
2. Ordinales: Tienen un orden implícito, pero las diferencias entre los valores no son medibles (Ejemplo: nivel de satisfacción: Bajo, Medio, Alto).

Ejemplo en Python:

```
import pandas as pd

df = pd.DataFrame({"Género": ["Masculino", "Femenino", "Masculino", "Otro"]})
print(df["Género"].value_counts()) # Cuenta la frecuencia de cada categoría
```

Ilustración 1 Ejemplo variables categóricas

Variables Cuantitativas

Representan cantidades numéricas. Pueden ser discretas o continuas.

Ejemplos:

- Edad en años (15, 22, 30, ...).
- Altura en cm (175, 182, 160, ...).
- Ingresos mensuales en dólares (1500, 2800, 3200, ...).

Tipos de Variables Cuantitativas:

1. Discretas: Toman valores enteros contables, sin decimales. (Ejemplo: número de hijos: 0, 1, 2, ...).
2. Continuas: Pueden tomar cualquier valor dentro de un rango, incluyendo decimales. (Ejemplo: temperatura: 36.5°C, 22.3°C).

Ejemplo en Python:

```
import numpy as np

# Variable Discreta
num_hijos = np.array([0, 1, 2, 3, 2, 1, 0, 3])
print("Promedio de hijos:", np.mean(num_hijos))

# Variable Continua
pesos = np.array([65.2, 70.1, 68.5, 72.3])
print("Peso máximo:", np.max(pesos))
```

Ilustración 2 Ejemplo variables cuantitativas

TABLAS DE FRECUENCIA

¿Qué es una Tabla de Frecuencia?

Una tabla de frecuencia es una herramienta estadística utilizada para organizar y resumir datos, mostrando cuántas veces aparece cada valor dentro de una variable. Ayuda a visualizar la distribución de los datos y a identificar patrones de frecuencia.

Tipos:

- Frecuencia absoluta: Número de veces que aparece cada categoría.
- Frecuencia relativa: Proporción de cada categoría en relación con el total.
- Frecuencia acumulada: Suma progresiva de las frecuencias.

Ejemplo en Python:

```
import pandas as pd

# Datos de ejemplo: Calificaciones de un grupo de estudiantes
datos = [7, 8, 6, 9, 7, 6, 8, 9, 7, 8, 7, 9, 6, 7, 8]

# Crear un DataFrame
df = pd.DataFrame(datos, columns=["Calificación"])

# Crear la tabla de frecuencias
tabla_frecuencia = df["Calificación"].value_counts().sort_index().to_frame("Frecuencia Absoluta")
tabla_frecuencia["Frecuencia Relativa"] = tabla_frecuencia["Frecuencia Absoluta"] / tabla_frecuencia["Frecuencia Absoluta"].sum()
tabla_frecuencia["Frecuencia Acumulada"] = tabla_frecuencia["Frecuencia Absoluta"].cumsum()
tabla_frecuencia["Frecuencia Relativa Acumulada"] = tabla_frecuencia["Frecuencia Relativa"].cumsum()

# Mostrar la tabla de frecuencias
print(tabla_frecuencia)
```

Ilustración 3 Ejemplo tabla de frecuencia

Interpretación de la Tabla

Calificación	Frecuencia Absoluta	Frecuencia Relativa	Frecuencia Acumulada	Frecuencia Relativa Acumulada
6	3	0.20	3	0.20
7	5	0.33	8	0.53
8	4	0.27	12	0.80
9	3	0.20	15	1.00

- ✓ Frecuencia Absoluta: La calificación 7 apareció 5 veces.
- ✓ Frecuencia Relativa: El 20% de los estudiantes obtuvo una calificación de 6.
- ✓ Frecuencia Acumulada: Hasta la calificación 8, ya se han registrado 12 estudiantes.
- ✓ Frecuencia Relativa Acumulada: El 80% de los estudiantes obtuvo una calificación de 8 o menor.

MEDIDAS DE TENDENCIA CENTRAL: MEDIA, MEDIANA Y MODA

Las medidas de tendencia central son estadísticas que nos permiten identificar valores representativos de un conjunto de datos. Estas medidas nos dan una idea de cuál es el valor típico o central en un conjunto de datos y son fundamentales para resumir la información y compararla entre diferentes grupos o conjuntos.

Media (Promedio)

La media es el valor promedio de un conjunto de datos. Se calcula sumando todos los valores y dividiendo entre el número total de elementos en el conjunto de datos. La media es muy útil cuando los datos son relativamente simétricos, pero puede ser sensible a los valores atípicos (outliers), ya que estos afectan considerablemente el promedio.

Fórmula:

$$\text{Media} = \frac{\sum X_i}{N}$$

Ilustración 4 Formula de la media

Donde:

- X_i son los valores individuales del conjunto de datos.
- N es el número total de elementos en el conjunto de datos

Para calcular la media, se utiliza la función `np.mean()` de NumPy. Esta función calcula el valor promedio de todos los elementos de un conjunto de datos.

Ejemplo en Python:

```
import numpy as np

# Conjunto de datos de ejemplo
datos = [3, 5, 7, 9, 11]

# Calcular la media utilizando numpy
media = np.mean(datos)

print(f"La media del conjunto de datos es: {media}")
```

Ilustración 5 Ejemplo en Python para la media

Explicación:

En este ejemplo, la función `np.mean()` suma todos los elementos del conjunto de datos [3, 5, 7, 9, 11] (que da 35) y lo divide entre el número total de elementos (5). El resultado es 7.0, que es el valor promedio del conjunto de datos.

Mediana

La mediana es el valor central de un conjunto de datos ordenados. Si el número de elementos es impar, la mediana será el valor que esté justo en el medio de la lista ordenada. Si el número de valores es par, la mediana se calcula como el promedio de los dos valores centrales. La mediana es especialmente útil cuando los datos están sesgados o contienen valores atípicos, ya que no se ve afectada por los extremos.

Caso especial: Si el número de valores es par, se toma el promedio de los dos valores centrales.



Para calcular la mediana, se utiliza la función `np.median()` de NumPy. Esta función ordena los datos y devuelve el valor central. Si hay un número par de elementos, se toma el promedio de los dos valores centrales.

Ejemplo en Python:

```
# Conjunto de datos de ejemplo
datos = [3, 5, 7, 9, 11]

# Calcular la mediana utilizando numpy
mediana = np.median(datos)

print(f"La mediana del conjunto de datos es: {mediana}")
```

Ilustración 6 Ejemplo en Python para la mediana

Explicación:

- En este caso, la función `np.median()` primero ordena los datos (aunque ya están ordenados: `[3, 5, 7, 9, 11]`). Como hay un número impar de elementos, devuelve el valor que está en la posición central, que es 7.
- Si los datos fueran `[3, 5, 7, 9]` (número par de elementos), la función tomaría el promedio de los dos valores centrales, es decir, el promedio de 5 y 7, que sería 6.0.

Moda

La moda es el valor que aparece con mayor frecuencia en un conjunto de datos. A diferencia de la media y la mediana, la moda puede no existir o puede tener más de un valor (cuando varios valores se repiten con la misma frecuencia). Es útil cuando se desean identificar las categorías o valores más frecuentes, especialmente en conjuntos de datos cualitativos.



Para calcular la moda, se utiliza la función stats.mode() de SciPy. Esta función devuelve el valor que aparece con mayor frecuencia en el conjunto de datos. Si hay varios valores con la misma frecuencia máxima, la función devuelve el primer valor encontrado.

Ejemplo en Python:

```
from scipy import stats

# Conjunto de datos de ejemplo
datos = [3, 5, 7, 5, 9, 5, 11]

# Calcular la moda utilizando scipy
moda = stats.mode(datos)

print(f"La moda del conjunto de datos es: {moda.mode[0]}")
```

Ilustración 7 Ilustración 6 Ejemplo en Python para la moda

Explicación:

En este caso, la función stats.mode() analiza los datos [3, 5, 7, 5, 9, 5, 11] y encuentra que el número 5 aparece más veces que cualquier otro valor (tres veces). Por lo tanto, la moda es 5.

MEDIDAS DE DISPERSIÓN: RANGO DE VARIACIÓN, VARIANZA Y DESVIACIÓN ESTÁNDAR

Las medidas de dispersión nos permiten entender la variabilidad de un conjunto de datos. Estas medidas indican cuán dispersos o concentrados están los valores alrededor de la media, proporcionando información valiosa sobre la distribución de los datos.

Rango de Variación

El rango de variación es una medida simple de dispersión que muestra la amplitud total de los valores en un conjunto de datos. Se calcula restando el valor mínimo del conjunto del valor máximo. Aunque es fácil de calcular, el rango puede verse afectado por los valores atípicos, por lo que no siempre refleja con precisión la dispersión.

Fórmula:

$$Rango = \text{Valor Máximo} - \text{Valor Mínimo}$$

Ilustración 8 Fórmula rango de variación

Ejemplo en Python:

```
# Conjunto de datos de ejemplo
datos = [3, 5, 7, 9, 11]

# Calcular el rango
rango = max(datos) - min(datos)

print(f"El rango de variación es: {rango}")
```

Ilustración 9 Ejemplo en Python de rango de variación

Explicación:

En este ejemplo, el valor máximo es 11 y el valor mínimo es 3. Por lo tanto, el rango es $11 - 3 = 8$, lo que indica que la variabilidad entre los valores es 8 unidades.

Varianza

La varianza es una medida de dispersión que indica cuánto se alejan los valores de un conjunto de datos respecto a la media. Se calcula como el promedio de las diferencias al cuadrado entre cada valor y la media. Una varianza alta indica que los valores están más dispersos, mientras que una varianza baja indica que están más concentrados alrededor de la media.

Fórmula para una población completa:

$$\sigma^2 = \frac{\sum(X_i - \mu)^2}{N}$$

Ilustración 10 Formula de varianza para población completa

Fórmula para una muestra (usando la Corrección de Bessel):

$$s^2 = \frac{\sum(X_i - \bar{X})^2}{N - 1}$$

Ilustración 11 Formula de varianza para muestra

Ejemplo en Python:

```
import numpy as np

# Conjunto de datos de ejemplo
datos = [3, 5, 7, 9, 11]

# Calcular la varianza poblacional (sin corrección)
varianza_poblacional = np.var(datos) # Para población

# Calcular la varianza muestral (con corrección de Bessel)
varianza_muestral = np.var(datos, ddof=1) # Para muestra con corrección de Bessel

# Imprimir los resultados
print("Varianza poblacional:", varianza_poblacional)
print("Varianza muestral:", varianza_muestral)
```

Ilustración 12 Ejemplo en Python de varianza

Explicación:

- Varianza poblacional: Se calcula usando `np.var(datos)`, donde se usa todo el conjunto de datos y se divide entre el número total de elementos.
- Varianza muestral: Se calcula usando `np.var(datos, ddof=1)`, lo que ajusta el cálculo para muestras al dividir entre $\eta - 1$ (grados de libertad), lo que corrige el sesgo en la estimación de la varianza.
- El código imprime ambas varianzas, permitiendo comparar la dispersión de los datos, ya sea para toda la población o para una muestra específica.

Desviación Estándar

La desviación estándar es la raíz cuadrada de la varianza y mide cuánto se alejan los valores de la media. Es más fácil de interpretar que la varianza, ya que está expresada en las mismas unidades que los datos originales. Una desviación estándar pequeña indica que los valores están cerca de la media, mientras que una desviación estándar grande indica una mayor dispersión.

Fórmula:

$$\sigma = \sqrt{\text{Varianza}}$$

Ilustración 13 Fórmula desviación estándar

Ejemplo en Python:

```
desviacion_poblacional = np.std(datos) # Para población
desviacion_muestral = np.std(datos, ddof=1) # Para muestra con corrección de Bessel
print("Desviación estándar poblacional:", desviacion_poblacional)
print("Desviación estándar muestral:", desviacion_muestral)
```

Ilustración 14 Ejemplo en Python de desviación estándar

Explicación:

- `np.std(datos)`: Calcula la desviación estándar poblacional, que mide la dispersión de todos los datos con respecto a la media.
- `np.std(datos, ddof=1)`: Calcula la desviación estándar muestral, aplicando la corrección de Bessel
- $(n - 1)$ para ajustar la estimación en caso de trabajar con una muestra.
- La desviación estándar es la raíz cuadrada de la varianza, y nos indica cuánto se alejan los valores del promedio en las mismas unidades que los datos originales.

POBLACIÓN, MUESTRA Y CORRECCIÓN DE BESEL

En estadística, muchas veces no es posible analizar a toda una población debido a restricciones de tiempo o recursos. En su lugar, se trabaja con una muestra, es decir, un subconjunto representativo de los datos. Sin embargo, cuando se calculan estadísticas como la varianza o la desviación estándar a partir de una muestra, es necesario aplicar la corrección de Bessel para evitar subestimar la variabilidad real de la población.

Población vs Muestra

- ▶ **Población:** Es el conjunto completo de datos sobre el cual se desea realizar un estudio. Puede incluir a todas las personas, objetos o eventos que cumplan con ciertos criterios.
- ▶ **Muestra:** Es un subconjunto de la población que se selecciona para análisis debido a limitaciones de tiempo, costo o accesibilidad. La muestra debe ser representativa para que los resultados puedan generalizarse a la población.

Ejemplo en Python:

```
import numpy as np

# Definir la población completa
poblacion = np.array([5, 10, 15, 20, 25, 30, 35, 40])

# Seleccionar una muestra aleatoria de 5 elementos sin reemplazo
muestra = np.random.choice(poblacion, size=5, replace=False)

# Imprimir la población y la muestra seleccionada
print("Población:", poblacion)
print("Muestra:", muestra)
```

Ilustración 15 Ejemplo en Python de población y muestra

Explicación:

- Se define un array `poblacion` con un conjunto de valores representando una población completa.
- `np.random.choice(poblacion, size=5, replace=False)` selecciona 5 elementos aleatorios de la población sin reemplazo, lo que significa que no se repiten valores en la muestra.
- Finalmente, se imprimen la población y la muestra seleccionada.

Corrección de Bessel

En estadística, la corrección de Bessel se usa al calcular la varianza y la desviación estándar de una muestra en lugar de toda la población. Se divide entre $N - 1$ en lugar de N para corregir la subestimación de la variabilidad de la población.

Fórmula para la varianza muestral con corrección de Bessel:

$$s^2 = \frac{\sum(X_i - \bar{X})^2}{N - 1}$$

Ilustración 16 Fórmula para la varianza muestral con corrección de Bessel

Ejemplo en Python

```
# Cálculo de la varianza con y sin corrección de Bessel
varianza_poblacional = np.var(muestra) # División entre N
varianza_muestral = np.var(muestra, ddof=1) # División entre N-1

print("Varianza poblacional:", varianza_poblacional)
print("Varianza muestral con corrección de Bessel:", varianza_muestral)
```

Ilustración 17 Ejemplo en Python de corrección de Bessel

Explicación del código:

- `np.var(muestra)`: Calcula la varianza sin la corrección de Bessel, dividiendo entre N.
- `np.var(muestra, ddof=1)`: Aplica la corrección de Bessel dividiendo entre N-1, proporcionando una estimación más precisa de la varianza poblacional.

MEDIDAS DE POSICIÓN (CUARTIL, QUINTIL, DECIL, PERCENTIL)

Las medidas de posición permiten dividir un conjunto de datos en partes iguales para analizar la distribución de los valores. Estas medidas son fundamentales en estadística para entender la dispersión y la tendencia de los datos en diferentes puntos de referencia.

Cuartiles

Dividen los datos en cuatro partes iguales (Q1, Q2, Q3).

- Q1 (25%): Primer cuartil, el 25% de los valores están por debajo de este punto.
- Q2 (50%): Mediana, el 50% de los valores están por debajo.
- Q3 (75%): Tercer cuartil, el 75% de los valores están por debajo.

Ejemplo en Python:

```
import numpy as np

# Datos de ejemplo
datos = np.array([5, 10, 15, 20, 25, 30, 35, 40])

# Cálculo de los cuartiles
Q1 = np.percentile(datos, 25)
Q2 = np.percentile(datos, 50) # Mediana
Q3 = np.percentile(datos, 75)

print("Primer cuartil (Q1):", Q1)
print("Mediana (Q2):", Q2)
print("Tercer cuartil (Q3):", Q3)
```

Ilustración 18 Ejemplo en Python para calcular cada cuartil

Explicación:

Se usa `np.percentile()` para calcular cada cuartil. Estos valores ayudan a entender cómo se distribuyen los datos y detectar posibles valores atípicos.

Quintiles, Deciles y Percentiles

- Quintiles: Dividen los datos en cinco partes iguales (Q1, Q2, Q3, Q4, Q5). Cada quintil representa un 20% de los datos.
- Deciles: Dividen los datos en diez partes iguales. Cada decil representa un 10% de los datos.
- Percentiles: Dividen los datos en cien partes iguales. Cada percentil representa un 1% de los datos.

Ejemplo en Python:

```
# Cálculo de quintiles, deciles y percentiles
quintil_2 = np.percentile(datos, 40) # Segundo quintil (40%)
decil_3 = np.percentile(datos, 30) # Tercer decil (30%)
percentil_90 = np.percentile(datos, 90) # Percentil 90

print("Segundo quintil (Q2 - 40%):", quintil_2)
print("Tercer decil (D3 - 30%):", decil_3)
print("Percentil 90:", percentil_90)
```

Ilustración 19 Ejemplo en Python de quintil, decil y percentil

Explicación: Se usa `np.percentile()` para calcular quintiles, deciles y percentiles. Estas medidas ayudan a analizar la distribución de los datos en más detalle y pueden ser útiles en estudios como análisis de ingresos, rendimiento académico o tiempos de respuesta en experimentos.

PUNTOS ATÍPICOS

Un punto atípico o outlier es un valor en un conjunto de datos que se aleja significativamente del resto de las observaciones. Es decir, es un dato que es inusualmente alto o bajo en comparación con los demás.

Los outliers pueden aparecer por varias razones:

- Errores en la medición o en la entrada de datos. Ejemplo: un sensor que registra una temperatura de 500°C cuando lo normal es entre 20°C y 30°C.
- Variabilidad natural de los datos. Ejemplo: en un maratón, la mayoría de los corredores terminan en 3-4 horas, pero hay unos pocos que terminan en menos de 2 horas.
- Eventos raros o excepcionales. Ejemplo: los ingresos de la mayoría de las personas en una empresa pueden estar entre \$1000 y \$5000, pero el CEO puede ganar \$50,000 al mes.

Detectar y manejar los outliers es importante porque pueden afectar los análisis estadísticos y los modelos de machine learning.

Método del Rango Intercuartílico (IQR) para detectar outliers

El rango intercuartílico (IQR) es una medida de dispersión que se usa para identificar valores atípicos.

Se basa en la diferencia entre el tercer cuartil (Q3) y el primer cuartil (Q1):

Fórmula

$$IQR = Q3 - Q1$$

$$\text{Límite Inferior} = Q1 - 1.5 \times IQR$$

$$\text{Límite Superior} = Q3 + 1.5 \times IQR$$

Ilustración 20 Fórmula IQR

Ejemplo en Python:

```
import numpy as np

# Datos de ejemplo con un posible outlier
datos = np.array([10, 12, 15, 14, 13, 102, 11, 12, 14, 13])

# Cálculo de cuartiles e IQR
Q1 = np.percentile(datos, 25)
Q3 = np.percentile(datos, 75)
IQR = Q3 - Q1

# Definir límites para detectar outliers
limite_inferior = Q1 - 1.5 * IQR
limite_superior = Q3 + 1.5 * IQR

# Identificar valores atípicos
outliers = datos[(datos < limite_inferior) | (datos > limite_superior)]

print("Datos originales:", datos)
print("Q1:", Q1, "| Q3:", Q3, "| IQR:", IQR)
print("Límite inferior:", limite_inferior, "| Límite superior:", limite_superior)
print("Outliers detectados:", outliers)
```

Ilustración 21 Ejemplo en Python de outlier

Explicación:

- Se define un conjunto de datos donde 102 es un posible outlier.
- Se calculan los cuartiles Q1 y Q3, y luego el IQR.
- Se establecen los límites de detección basados en la regla de $1.5 \times \text{IQR}$.
- Se identifican los valores fuera de estos límites como outliers.

Este método es muy utilizado en análisis de datos para limpiar conjuntos de datos y mejorar la precisión de los modelos estadísticos y de Machine Learning.

GRÁFICOS Y DIAGRAMAS: HISTOGRAMA Y BOXPLOT

Histograma

Un histograma es un gráfico que muestra la distribución de frecuencias de una variable numérica. Se utiliza para representar la distribución de datos agrupados en intervalos o "bins", proporcionando una visión clara de la forma de la distribución, como si es simétrica, sesgada, o tiene distribuciones normales o anormales.

Los histogramas son útiles para observar la dispersión, la tendencia central, y detectar patrones o anomalías, como la presencia de valores atípicos. El eje x representa los intervalos de los datos, mientras que el eje y muestra la frecuencia o el número de observaciones que caen dentro de cada intervalo.

Ejemplo en Python:

```
import matplotlib.pyplot as plt

plt.hist(datos, bins=5, edgecolor="black")
plt.xlabel("Valores")
plt.ylabel("Frecuencia")
plt.title("Histograma de Datos")
plt.show()
```

Ilustración 22 Ejemplo histograma

Boxplot (Diagrama de Caja)

Un boxplot, o diagrama de caja y bigotes, es una representación gráfica que resume la distribución de un conjunto de datos numéricos. Es una herramienta clave en la estadística descriptiva, ya que permite visualizar de manera rápida la dispersión de los datos, su simetría y la presencia de valores atípicos (outliers).

¿Para qué se usa un Boxplot?

- Identificar la dispersión de los datos: Permite ver qué tan dispersos o concentrados están los valores.
- Detectar valores atípicos (outliers): Señala valores que se alejan significativamente del resto de los datos.
- Comparar distribuciones: Ideal para comparar varias categorías de una variable numérica.
- Evaluar simetría y sesgo: Ayuda a ver si los datos están sesgados hacia un lado.

Elementos del Boxplot

- Mediana (línea dentro de la caja): Valor central de los datos.
- Primer cuartil (Q1, borde inferior de la caja): 25% de los datos son menores que este valor.
- Tercer cuartil (Q3, borde superior de la caja): 75% de los datos son menores que este valor.
- Rango Intercuartílico (IQR): Diferencia entre Q3 y Q1, representa la dispersión del 50% central de los datos.
- Bigotes (líneas que salen de la caja): Representan los valores dentro del rango aceptable ($Q1 - 1.5 * IQR$ y $Q3 + 1.5 * IQR$).
- Outliers (puntos fuera de los bigotes): Valores extremadamente altos o bajos en comparación con el resto.



Ejemplo en Python:

```
import seaborn as sns
import matplotlib.pyplot as plt

# Suponiendo que 'datos' es una lista o un array de datos
sns.boxplot(x=datos)

# Títulos y etiquetas
plt.title("Boxplot de Datos")

# Mostrar gráfico
plt.show()
```

Ilustración 23 Ejemplo boxplot

EJERCICIO PRÁCTICO: IDENTIFICACIÓN Y ANÁLISIS DE OUTLIERS EN UN CONJUNTO DE DATOS

En este ejercicio práctico, los alumnos aprenderán a identificar y analizar outliers (valores atípicos) en un conjunto de datos utilizando Python. Los outliers son observaciones que se desvían significativamente del resto de los datos y pueden afectar el análisis estadístico. Aprender a detectarlos y manejarlos es una habilidad esencial en ciencia de datos.

Paso 1: Importar Librerías

Instrucción: Importa las librerías necesarias para el análisis de datos y visualización.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Ilustración 24 Importación de librerías

Paso 2: Crear o Cargar un Conjunto de Datos

Instrucción: Crea un conjunto de datos simple o carga uno existente. Para este ejemplo, usaremos datos simulados.

```
# Crear un conjunto de datos con outliers
datos = {
    'Valores': [10, 12, 12, 13, 12, 14, 13, 15, 14, 12, 100, 11, 13, 12, 14, 13, 15, 14, 12, 11]
}

df = pd.DataFrame(datos)
print(df)
```

Ilustración 25 Creación de datos

Paso 3: Visualizar los Datos

Instrucción: Usa un gráfico de caja (boxplot) para visualizar la distribución de los datos y detectar outliers.

```
# Crear un boxplot
sns.boxplot(x=df['Valores'])
plt.title('Boxplot de Valores')
plt.show()
```

Ilustración 26 Visualización de datos

Interpretación del gráfico creado:

- Los puntos fuera de los "bigotes" del boxplot son considerados outliers.
- En este caso, el valor 100 es claramente un outlier.

Paso 4: Identificar Outliers Usando el Rango Intercuartílico (IQR)

Instrucción: Calcula el rango intercuartílico (IQR) y usa este valor para identificar outliers.

Cálculo del IQR:

- Q1 (Primer Cuartil): El valor que separa el 25% inferior de los datos.
- Q3 (Tercer Cuartil): El valor que separa el 75% inferior de los datos.
- IQR: $IQR = Q3 - Q1$.

Limites:

- Límite inferior: $Q1 - 1.5 \times IQR$.
- Límite superior: $Q3 + 1.5 \times IQR$.

```
# Calcular Q1, Q3 e IQR
Q1 = df['Valores'].quantile(0.25)
Q3 = df['Valores'].quantile(0.75)
IQR = Q3 - Q1

# Definir límites para outliers
limite_inferior = Q1 - 1.5 * IQR
limite_superior = Q3 + 1.5 * IQR

# Identificar outliers
outliers = df[(df['Valores'] < limite_inferior) | (df['Valores'] > limite_superior)]
print("Outliers identificados:")
print(outliers)
```

Ilustración 27 Cálculos de IQR

Salida:

```
Outliers identificados:
Valores
10      100
```

Ilustración 28 Salida

Paso 5: Analizar los Outliers

Instrucción: Analiza los outliers para determinar si son errores o valores válidos.

- Errores: Si son errores (por ejemplo, un valor mal registrado), puedes eliminarlos o corregirlos.
- Valores Válidos: Si son valores válidos pero atípicos, decide cómo manejarlos (por ejemplo, transformar los datos o usar métodos robustos).

Ejemplo:

En este caso, el valor 100 parece ser un error o un valor extremo que no es representativo de la distribución normal de los datos.

Paso 6: Manejar los Outliers

Instrucción: Decide cómo manejar los outliers. Algunas opciones comunes son:

- Eliminarlos: Si son errores.
- Transformar los Datos: Aplicar una transformación (por ejemplo, logarítmica) para reducir el impacto de los outliers.
- Usar Métodos Robustos: Utilizar técnicas estadísticas que sean menos sensibles a los outliers.

Eliminación de outliers en python

```
# Eliminar outliers
df_sin_outliers = df[(df['Valores'] >= limite_inferior) & (df['Valores'] <= limite_superior)]
print("Datos sin outliers:")
print(df_sin_outliers)
```

Ilustración 29 Eliminación de outliers en Python



Salida:

```
Datos sin outliers:  
  Valores  
  0      10  
  1      12  
  2      12  
  3      13  
  4      12  
  5      14  
  6      13  
  7      15  
  8      14  
  9      12  
 11     11  
 12     13  
 13     12  
 14     14  
 15     13  
 16     15  
 17     14  
 18     12  
 19     11
```

Ilustración 30 Salida

Paso 7: Visualizar los Datos sin Outliers

Instrucción: Crea un nuevo boxplot para visualizar los datos sin outliers.

```
# Crear un boxplot sin outliers  
sns.boxplot(x=df_sin_outliers['Valores'])  
plt.title('Boxplot de Valores sin Outliers')  
plt.show()
```

Ilustración 31 Código para crear un boxplot

Interpretación:

El nuevo boxplot debe mostrar una distribución más compacta y sin valores atípicos.



Conclusión del Ejercicio

- Aprendiste a identificar y analizar outliers en un conjunto de datos utilizando el rango intercuartílico (IQR).
- Practicaste la visualización de datos con boxplots y el manejo de outliers.
- Este proceso es esencial para garantizar la calidad de los datos y la validez de los análisis estadísticos.