

✓ Pre-procesamiento de Datos

El preprocesamiento es esencial para mejorar la calidad de los datos antes del modelado.

a. Limpieza de Datos:

Detección y tratamiento de valores nulos (NaN).

Detección de datos duplicados o inconsistentes.

Corrección de errores tipográficos o de codificación.

```
df.dropna(inplace=True) # Elimina filas con valores nulos
df.dropna(subset=['Turno'], inplace=True)
df.duplicated().sum() # Número de duplicados
df['columna'] = df['columna'].str.strip().str.upper() # Limpieza de strings
```

✓ Base de datos de atención a clientes en caja de compensacion familiar

```
1 # Conectarse a Google Drive para poder acceder a archivos almacenados allí
2 from google.colab import drive
3 drive.mount('/gdrive')
4
5 # Importar la librería pandas, que se usa para manipular y analizar datos en forma de tablas
6 import pandas as pd
7
8 # Leer el archivo CSV (separado por punto y coma) desde Google Drive.
9 # Se especifica la codificación 'ISO-8859-1' para evitar errores por caracteres especiales.
10 df = pd.read_csv("/gdrive/My Drive/METODOS_Y_MODELOS/4_Cuarta_Sesion/Observaciones_Turnos.csv", sep=";", encoding='ISO-8859-1')
11
12 df.head(10)
```

 [Mostrar salida oculta](#)

```
1 # Eliminar las filas donde la columna 'T.Atencion' tenga datos vacíos o nulos
2 df.dropna(subset=['T.Atencion'], inplace=True)
3
4 # Reiniciar el índice del DataFrame después de eliminar filas (para que sea consecutivo)
5 df.reset_index(drop=True, inplace=True)
6
7 # Asegurarse de que la columna 'T.Atencion' está en formato texto (string)
8 df['T.Atencion'] = df['T.Atencion'].astype(str)
9
10 # Convertir la columna 'T.Atencion' al tipo de dato 'timedelta', que representa duraciones de tiempo
11 df['T.Atencion'] = pd.to_timedelta(df['T.Atencion'])
12
13 # Crear una nueva columna llamada 'Minutos' que convierte la duración de tiempo en minutos (como número decimal)
14 df['Minutos'] = df['T.Atencion'].dt.total_seconds() / 60
15
16 # Ver la columna 'Funcionario', que contiene los nombres de los funcionarios
17 df['Funcionario']
18
19 # Mostrar una lista de los diferentes funcionarios sin repetirlos
20 df['Funcionario'].unique()
21
22 # Filtrar solo los valores de la columna 'Minutos' para el funcionario 'MLPULIDOP'
23 df1 = df[df['Funcionario'] == 'MLPULIDOP']['Minutos']
24
25 # Mostrar estadísticas descriptivas (como media, desviación estándar, mínimo, máximo, etc.) del tiempo de atención en minutos
26 df1.describe()
27
```



	Minutos
count	686.000000
mean	15.246088
std	33.110063
min	0.000000
25%	7.266667
50%	12.700000
75%	19.216667
max	837.733333

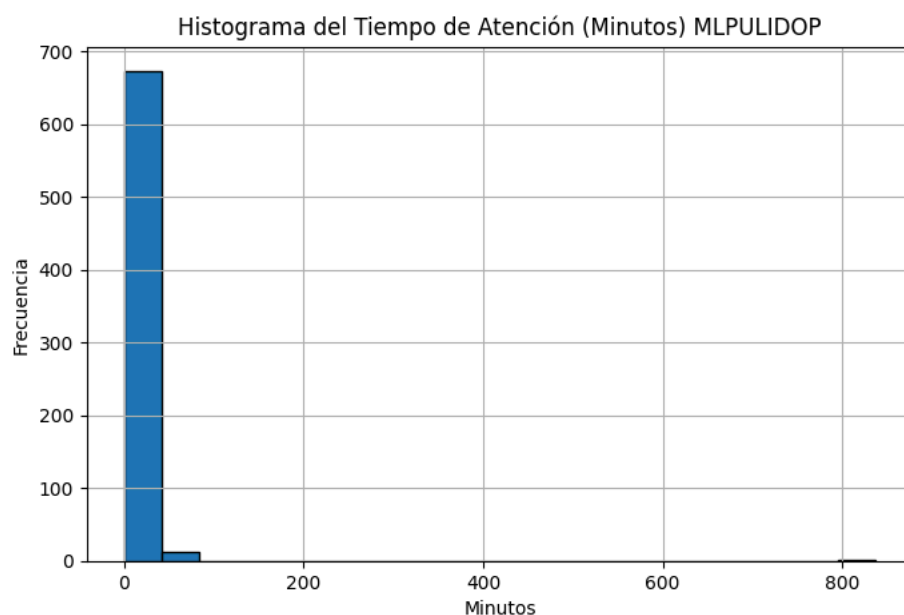
dtype: float64

✓ 1. Histograma – Distribución de los tiempos de atención

```

1 import matplotlib.pyplot as plt
2
3 plt.figure(figsize=(8, 5))
4 plt.hist(df1, bins=20, edgecolor='black')
5 plt.title('Histograma del Tiempo de Atención (Minutos) MLPULIDOP')
6 plt.xlabel('Minutos')
7 plt.ylabel('Frecuencia')
8 plt.grid(True)
9 plt.show()
10

```

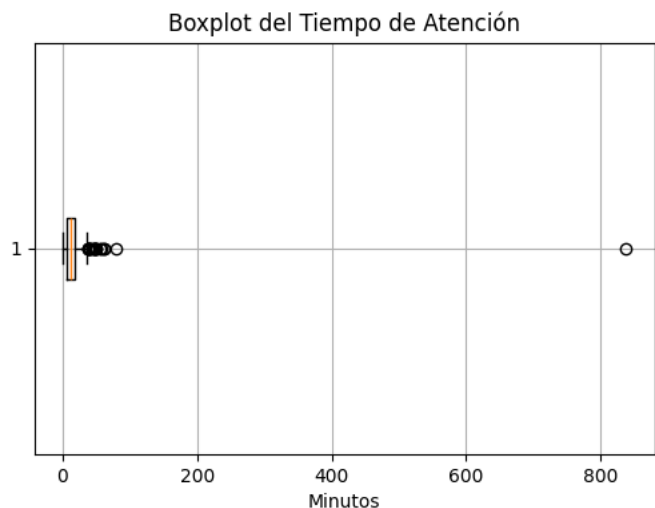


✓ 2. Boxplot – Visualización de valores atípicos

```

1 import matplotlib.pyplot as plt
2
3 plt.figure(figsize=(6, 4))
4 plt.boxplot(df1, vert=False)
5 plt.title('Boxplot del Tiempo de Atención')
6 plt.xlabel('Minutos')
7 plt.grid(True)
8 plt.show()

```



✓ 3. Descripción estadística + Identificación de outliers (IQR) (opcional)

Identificación de Valores Atípicos con el Rango Intercuartílico (IQR)

El **rango intercuartílico (IQR)** es una medida estadística que permite identificar valores atípicos (también llamados *outliers*) en un conjunto de datos.

¿Qué es el IQR?

El IQR es la diferencia entre el tercer cuartil (Q3) y el primer cuartil (Q1):

$$\text{IQR} = Q3 - Q1$$

- **Q1** (Primer cuartil): el valor por debajo del cual está el 25% de los datos.
- **Q3** (Tercer cuartil): el valor por debajo del cual está el 75% de los datos.

¿Cómo se detectan valores atípicos?

Un valor es considerado atípico si está **muy por debajo o muy por encima del rango típico de los datos**, es decir, fuera de los siguientes límites:

- **Límite inferior** = $Q1 - 1.5 \times \text{IQR}$
- **Límite superior** = $Q3 + 1.5 \times \text{IQR}$

Los valores que están por fuera de ese rango se consideran **outliers**.

¿Por qué es importante identificarlos?

Los valores atípicos pueden:

- Representar errores en la recolección de datos.
- Indicar casos especiales que requieren atención.
- Afectar el promedio y otros análisis estadísticos.

⚠ *No siempre se eliminan los outliers. Primero se analizan para decidir si son errores o si contienen información relevante.*

```
1 # Calcular valores límite para detectar outliers con el
2 # método del rango intercuartílico (IQR)
3
4 Q1 = df1.quantile(0.25)
5 print('Q1', Q1)
6
7 Q3 = df1.quantile(0.75)
8 IQR = Q3 - Q1
9 print('IQR', IQR)
10
11 # Límites
12 limite_inferior = Q1 - 1.5 * IQR
13 print(limite_inferior)
14 limite_superior = Q3 + 1.5 * IQR
```

```

15
16 # Mostrar outliers
17 outliers = df1[(df1 < limite_inferior) | (df1 > limite_superior)]
18 print("Valores atípicos (outliers):")
19 print(outliers.sort_values())
20 print('Minimo:', outliers.min())

```

```

↗ Q1 8.533333333333333
  IQR 10.283333333333333
  -6.8916666666666675
  Valores atípicos (outliers):
  Series([], Name: Minutos, dtype: float64)
  Minimo: nan

```

```

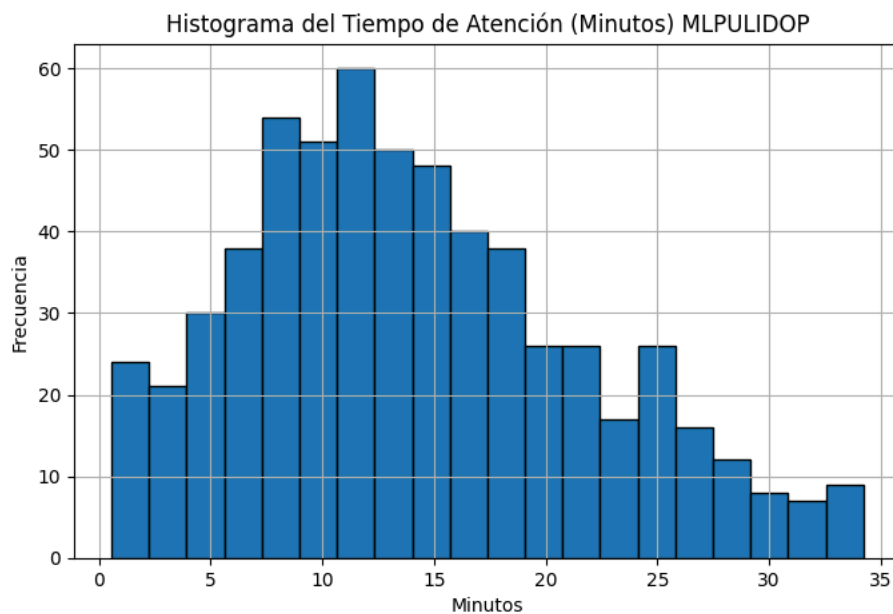
1 # Filtrar los datos eliminando los valores atípicos
2 df1 = df1[(df1 >= limite_inferior) & (df1 <= limite_superior)]
3 df1 = df1[(df1 >= 0.5) & (df1 <= outliers.min())]

```

```

1 import matplotlib.pyplot as plt
2
3 plt.figure(figsize=(8, 5))
4 plt.hist(df1, bins=20, edgecolor='black')
5 plt.title('Histograma del Tiempo de Atención (Minutos) MLPULIDOP')
6 plt.xlabel('Minutos')
7 plt.ylabel('Frecuencia')
8 plt.grid(True)
9 plt.show()
10

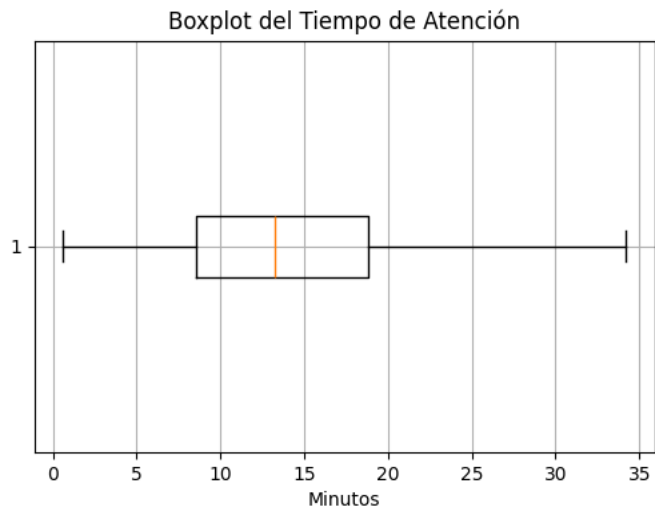
```



```

1 import matplotlib.pyplot as plt
2
3 plt.figure(figsize=(6, 4))
4 plt.boxplot(df1, vert=False)
5 plt.title('Boxplot del Tiempo de Atención')
6 plt.xlabel('Minutos')
7 plt.grid(True)
8 plt.show()

```



```
1 df1.describe()
```



	Minutos
count	601.000000
mean	14.168747
std	7.622382
min	0.566667
25%	8.533333
50%	13.250000
75%	18.816667
max	34.216667

dtype: float64

✓ Ajuste de distribuciones de probabilidad a datos reales

Una parte fundamental del análisis estadístico es conocer **qué distribución de probabilidad describe mejor nuestros datos**. Esto nos permite hacer inferencias más precisas, simulaciones y predicciones.

Objetivo

Queremos encontrar **cuál de las distribuciones de probabilidad comunes (Normal, Exponencial, Gamma, etc.) se ajusta mejor** a los tiempos de atención (en minutos) **sin valores atípicos**.

¿Cómo se hace?

1. **Ajustamos varias distribuciones** a nuestros datos usando `scipy.stats`.
2. Usamos la **prueba de Kolmogorov-Smirnov (KS)** para evaluar qué tan bien se ajusta cada distribución.
3. Elegimos la distribución que tenga el **mayor valor p (p-value)**, lo cual indica que nuestros datos podrían haber sido generados por esa distribución (no se rechaza la hipótesis nula).

⚠ Interpretación del p-value:

- Si el p-value es **mayor a 0.05**, no se rechaza la hipótesis de que los datos provienen de esa distribución.
- Cuanto **mayor sea el p-value**, mejor es el ajuste.

Distribuciones evaluadas:

- `norm`: Distribución normal
- `expon`: Exponencial
- `gamma`: Gamma
- `lognorm`: Log-normal


- beta: Beta
- weibull_min: Weibull

Visualización

Una vez seleccionada la mejor distribución, se puede graficar el **histograma de los datos reales** y superponer la **curva de la función de densidad (PDF)** ajustada, para visualizar qué tan bien se adapta.

Esto nos ayuda a comunicar los resultados de forma clara y visual.

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import scipy.stats as stats
4
5 datos = df1.values
6
7 # 1. Normalidad
8 param_norm = stats.norm.fit(datos)
9 ks_norm = stats.kstest(datos, 'norm', args=param_norm)
10 ks_norm
11
```

 KstestResult(statistic=np.float64(0.06213093270599068), pvalue=np.float64(0.018488449445507982), statistic_location=np.float64(12.083333333333334), statistic_sign=np.int8(1))

↯ Interpretación del Resultado de la Prueba KS (KstestResult)

Cuando aplicas la prueba de Kolmogorov–Smirnov con `scipy.stats.kstest`, el resultado es un objeto `KstestResult` con varios componentes. A continuación, se explica qué significa cada uno:

Campo	Qué significa	Explicación
statistic	D	Es el estadístico KS: la máxima diferencia entre la CDF empírica y la CDF teórica.
pvalue	Valor-p	Mide la significancia estadística . Si ($p < 0.05$), se rechaza la hipótesis nula , lo que sugiere que los datos no siguen la distribución teórica.
statistic_location	Ubicación de la diferencia	Es el valor de los datos donde ocurre la diferencia máxima entre ambas funciones acumuladas.
statistic_sign	Sentido de la diferencia	Si es 1 , la CDF empírica está por encima de la teórica. Si es -1 , está por debajo .

Ejemplo de resultado interpretado con la distribución Normal:

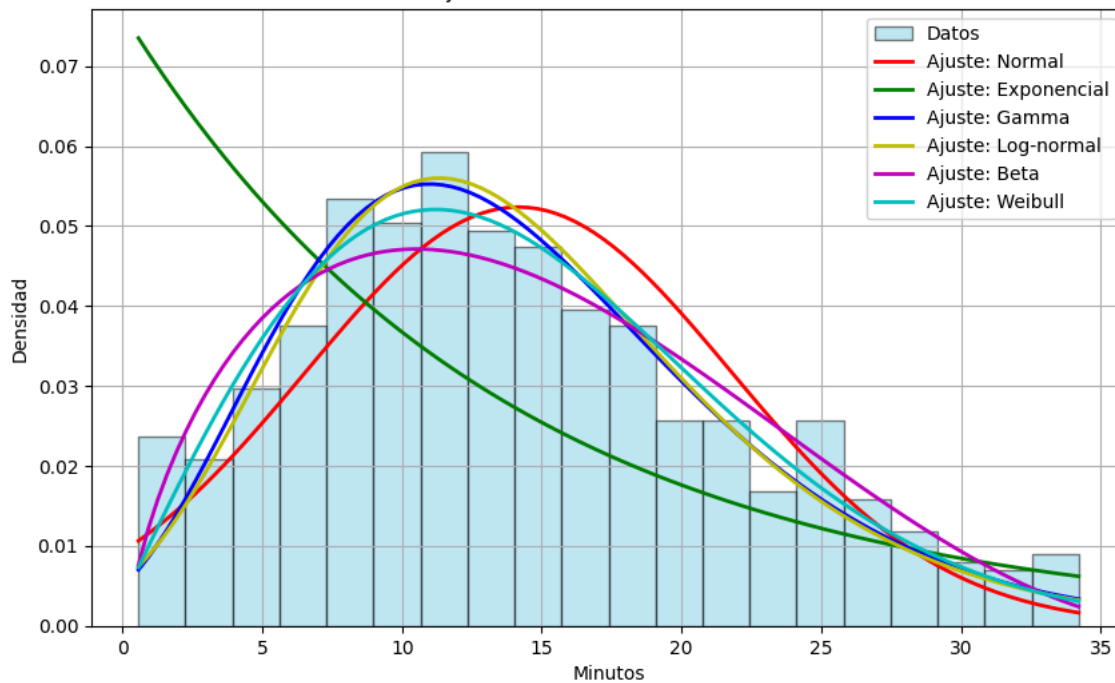
```
KstestResult(
    statistic=np.float64(0.06213093270599068),
    pvalue=np.float64(0.018488449445507982),
    statistic_location=np.float64(12.083333333333334),
    statistic_sign=np.int8(1)
)
```

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import scipy.stats as stats
4
5 # Convertir los datos a un array de NumPy
6 datos = df1.values
7
8 # -----
9 # 1. Ajuste de distribución Normal
10 params_norm = stats.norm.fit(datos)
11 ks_norm = stats.kstest(datos, 'norm', args=params_norm)
12
13 # 2. Ajuste de distribución Exponencial
14 params_expon = stats.expon.fit(datos)
15 ks_expon = stats.kstest(datos, 'expon', args=params_expon)
16
17 # 3. Ajuste de distribución Gamma
18 params_gamma = stats.gamma.fit(datos)
19 ks_gamma = stats.kstest(datos, 'gamma', args=params_gamma)
20
21 # 4. Ajuste de distribución Log-normal
22 params_lognorm = stats.lognorm.fit(datos)
23 ks_lognorm = stats.kstest(datos, 'lognorm', args=params_lognorm)
```

```
24
25 # 5. Ajuste de distribución Beta
26 params_beta = stats.beta.fit(datos)
27 ks_beta = stats.kstest(datos, 'beta', args=params_beta)
28
29 # 6. Ajuste de distribución Weibull
30 params_weibull = stats.weibull_min.fit(datos)
31 ks_weibull = stats.kstest(datos, 'weibull_min', args=params_weibull)
32
33 print("Resultados de la prueba KS (p-value):")
34 print(f"Normal      : {ks_norm}")
35 print(f"Exponencial : {np.round(ks_expon, 2)}")
36 print(f"Exponencial : {ks_expon}")
37 print(f"Gamma        : {ks_gamma}")
38 print(f"Log-normal   : {ks_lognorm}")
39 print(f"Beta         : {ks_beta}")
40 print(f"Weibull      : {ks_weibull}")
41
42
43 # Crear el rango de valores para graficar la curva
44 x = np.linspace(min(datos), max(datos), 100)
45 pdf_norm = stats.norm.pdf(x, *params_norm)
46 pdf_expon = stats.expon.pdf(x, *params_expon)
47 pdf_gamma = stats.gamma.pdf(x, *params_gamma)
48 pdf_lognorm = stats.lognorm.pdf(x, *params_lognorm)
49 pdf_beta = stats.beta.pdf(x, *params_beta)
50 pdf_weibull = stats.weibull_min.pdf(x, *params_weibull)
51
52
53 # Graficar el histograma de los datos con la curva ajustada
54 plt.figure(figsize=(10, 6))
55 plt.hist(datos, bins=20, density=True, alpha=0.5, color='skyblue', edgecolor='black', label='Datos')
56 plt.plot(x, pdf_norm, 'r-', lw=2, label='Ajuste: Normal')
57 plt.plot(x, pdf_expon, 'g-', lw=2, label='Ajuste: Exponencial')
58 plt.plot(x, pdf_gamma, 'b-', lw=2, label='Ajuste: Gamma')
59 plt.plot(x, pdf_lognorm, 'y-', lw=2, label='Ajuste: Log-normal')
60 plt.plot(x, pdf_beta, 'm-', lw=2, label='Ajuste: Beta')
61 plt.plot(x, pdf_weibull, 'c-', lw=2, label='Ajuste: Weibull')
62 plt.title('Ajuste de la distribución Normal')
63 plt.xlabel('Minutos')
64 plt.ylabel('Densidad')
65 plt.legend()
66 plt.grid(True)
67 plt.show()
```

Resultados de la prueba KS (p-value):
 Normal : KstestResult(statistic=np.float64(0.06213093270599068), pvalue=np.float64(0.018488449445507982), statistic_loc
 Exponencial : [0.21 0.]
 Exponencial : KstestResult(statistic=np.float64(0.20658491689116723), pvalue=np.float64(5.677039943311933e-23), statistic_lo
 Gamma : KstestResult(statistic=np.float64(0.026622613106001114), pvalue=np.float64(0.7774572712828338), statistic_loca
 Log-normal : KstestResult(statistic=np.float64(0.029213871686767034), pvalue=np.float64(0.672928437136046), statistic_locat
 Beta : KstestResult(statistic=np.float64(0.03619985909390233), pvalue=np.float64(0.4008185953537837), statistic_locat
 Weibull : KstestResult(statistic=np.float64(0.026673589015019594), pvalue=np.float64(0.775481692931755), statistic_locat

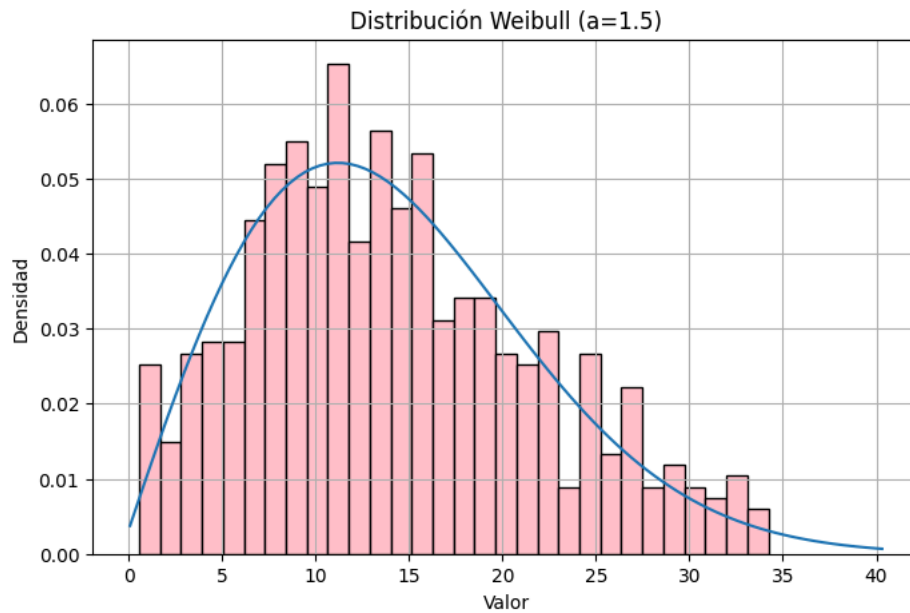
Ajuste de la distribución Normal



```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 c, loc, scale = stats.weibull_min.fit(datos)
5
6 # Paso 1: generar con np.random.weibull (loc=0, scale=1)
7 x_base = np.random.weibull(c, size=1000)
8
9 # Paso 2: aplicar transformación
10 x_simulado = loc + scale * x_base
11
12 # Crear eje x ordenado para la curva teórica (PDF)
13 x = np.linspace(min(x_simulado), max(x_simulado), 100)
14 pdf_weibull = stats.weibull_min.pdf(x, c, loc=loc, scale=scale)
15
16 plt.figure(figsize=(8, 5))
17 plt.hist(datos, bins=30, density=True, color='pink', edgecolor='black')
18 plt.plot(x, pdf_weibull)
19 plt.title("Distribución Weibull (a=1.5)")
20 plt.xlabel("Valor")
21 plt.ylabel("Densidad")
22 plt.grid(True)
23 plt.show()
24

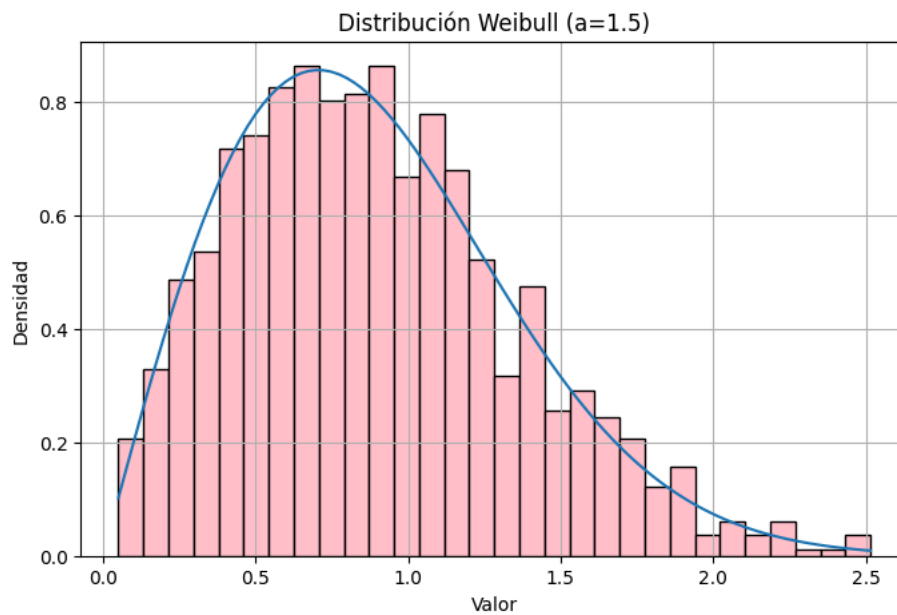
```

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 data = np.random.weibull(a, n)
5 x = np.linspace(min(data), max(data), 100)
6
7
8
9 plt.figure(figsize=(8, 5))
10
11 plt.hist(data, bins=30, density=True, color='pink', edgecolor='black')
12 plt.plot(x, pdf_weibull)
13 plt.title("Distribución Weibull ( $a=1.5$ )")
14 plt.xlabel("Valor")
15 plt.ylabel("Densidad")
16 plt.grid(True)
17 plt.show()
18

```



✓ QQ-Plot (Quantile-Quantile Plot)

El **QQ-Plot** es una herramienta gráfica que compara los **cuantiles teóricos** de una distribución de probabilidad con los **cuantiles observados** de nuestros datos.

¿Cómo se interpreta?

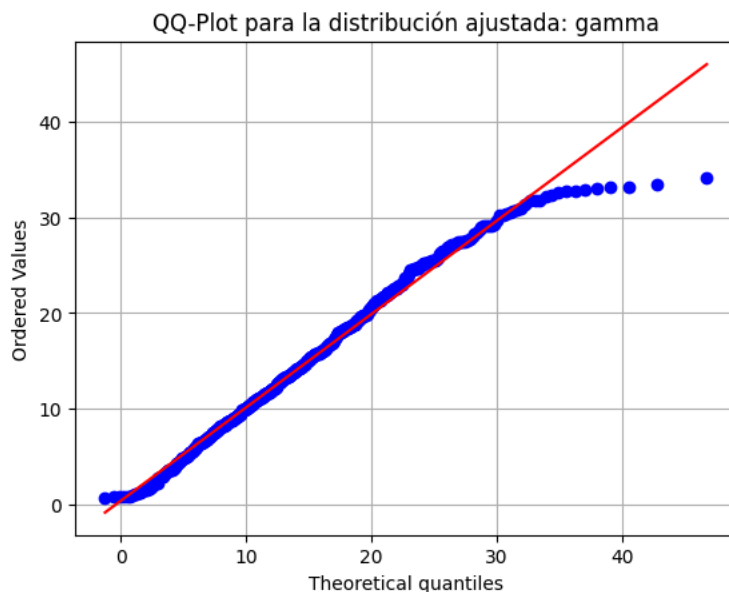
- Si los puntos se alinean aproximadamente sobre una línea recta diagonal, quiere decir que **los datos se ajustan bien a la distribución teórica**.
- Si los puntos **se desvían mucho de la línea**, entonces **la distribución no es un buen modelo para los datos**.

¿Para qué sirve?

- Visualiza de forma clara qué tan bien se ajustan los datos a una distribución específica.
- Es una herramienta complementaria a la prueba de Kolmogorov-Smirnov.

En nuestro caso, aplicamos el QQ-Plot a la **mejor distribución ajustada según el p-value**, para verificar gráficamente su adecuación.

```
1 import scipy.stats as stats
2 import matplotlib.pyplot as plt
3
4 # Elegimos la mejor distribución obtenida previamente
5 mejor_dist_name, mejor_info = mejores[0]
6 mejor_dist = getattr(stats, mejor_dist_name)
7 params = mejor_info['params']
8
9 # Generamos el QQ-Plot
10 stats.probplot(datos, dist=mejor_dist_name, sparams=params, plot=plt)
11 plt.title(f"QQ-Plot para la distribución ajustada: {mejor_dist_name}")
12 plt.grid(True)
13 plt.show()
```



✓ Pruebas estadísticas para evaluar el ajuste de una distribución

Cuando tenemos un conjunto de datos y queremos saber si **se ajustan a una distribución teórica** (por ejemplo, Normal, Gamma, Exponencial, etc.), existen varias pruebas estadísticas que nos ayudan a comprobar ese "fit":

1. Kolmogorov-Smirnov (KS)

Compara la función de distribución acumulada **empírica (ECDF)** con la **teórica (CDF)**.

- Estadístico: máxima diferencia entre ambas curvas.
- Sensible en la zona central.
- Requiere que los parámetros de la distribución estén definidos de antemano.

```
from scipy import stats
stats.kstest(datos, 'norm', args=(media, desviacion))
```

Pasos de la Prueba de Kolmogorov–Smirnov (KS)

Objetivo:

Evaluar si un conjunto de datos se ajusta a una distribución teórica.

Pasos:

1. **Ordenar los datos** de menor a mayor.

2. **Calcular la CDF empírica**

CDF = Cumulative Distribution Function

En español: Función de Distribución Acumulada.

- Es la proporción de datos que son menores o iguales a cada valor.
- Representa la distribución acumulada observada.

3. **Calcular la CDF teórica**

- Se obtiene usando la distribución que se está evaluando (por ejemplo: `stats.norm.cdf` si es distribución normal).
- Representa la distribución acumulada esperada bajo la hipótesis nula.

4. **Comparar ambas CDFs**

- Se calcula la **diferencia máxima** entre la CDF empírica y la CDF teórica:

$$D = \max |F_{\text{empírica}}(x) - F_{\text{teórica}}(x)|$$

5. **Obtener el p-valor**

- Asociado al estadístico (D).
 - Se interpreta en función del nivel de significancia (por ejemplo, 0.05).
-

Conclusión: