

Actividad Integradora

Ozner Axel Leyva Mariscal - A01742377

```
In [ ]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from scipy.stats import boxcox, yeojohnson
from statsmodels.stats.diagnostic import normal_ad
```

```
In [ ]: df = pd.read_csv('food_data_g.csv', index_col=0)
df = df["Monounsaturated Fats"].dropna()
df
```

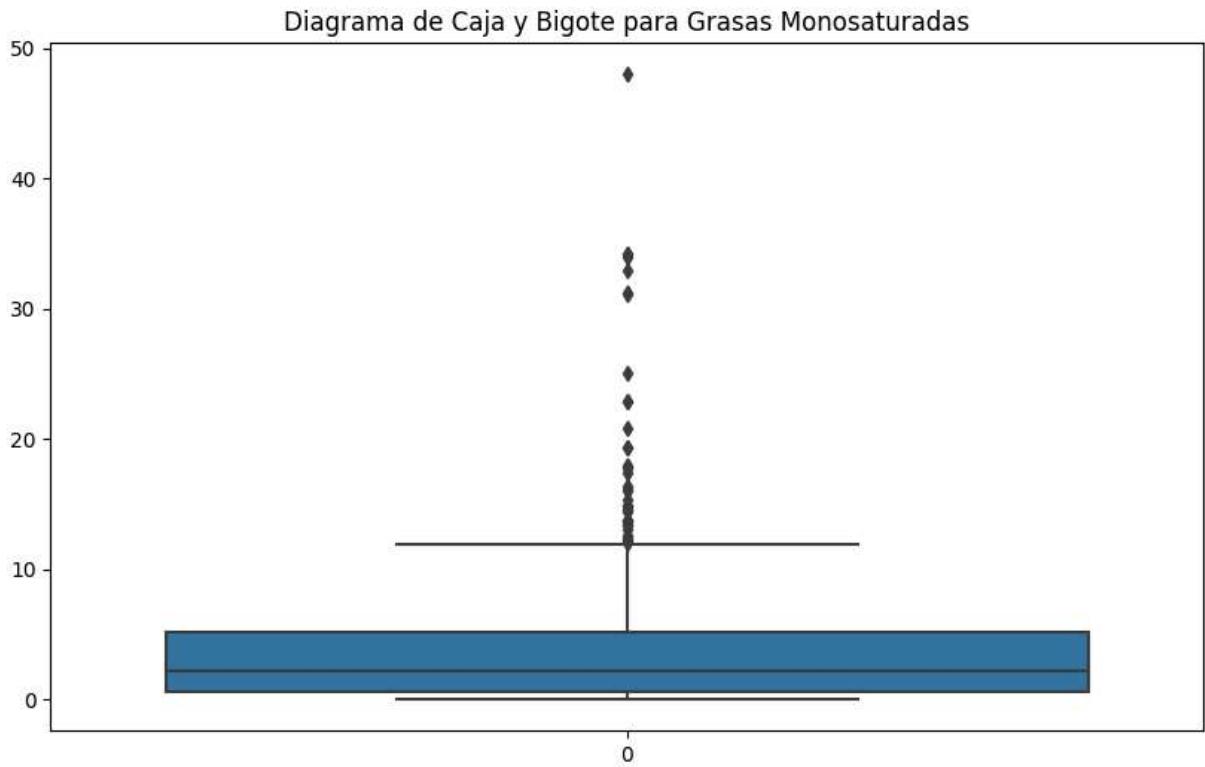
```
Out[ ]: 0      1.300
1      4.900
2      0.900
3      0.500
4      0.600
...
546    2.800
547    1.600
548    0.006
549    0.400
550    4.300
Name: Monounsaturated Fats, Length: 551, dtype: float64
```

Punto 1. Análisis descriptivo de la variable

Análisis de datos atípicos

Graficar el diagrama de caja y bigote

```
In [ ]: plt.figure(figsize=(10, 6))
sns.boxplot(df)
plt.title('Diagrama de Caja y Bigote para Grasas Monosaturadas')
plt.show()
```



Calcula las principales medidas

```
In [ ]: # Calcular medidas estadísticas
Q1 = np.percentile(df, 25)
Q2 = np.percentile(df, 50)
Q3 = np.percentile(df, 75)
IQR = Q3 - Q1
media = np.mean(df)
std_dev = np.std(df)

print(f'Cuartil 1 (Q1): {Q1}')
print(f'Mediana: {Q2}')
print(f'Cuartil 3 (Q3): {Q3}')
print(f'Rango Intercuartílico (IQR): {IQR}')
print(f'Media: {media}')
print(f'Desviación estándar: {std_dev}'')
```

Cuartil 1 (Q1): 0.6
 Mediana: 2.2
 Cuartil 3 (Q3): 5.15
 Rango Intercuartílico (IQR): 4.550000000000001
 Media: 4.00156442831216
 Desviación estándar: 5.535578198505148

Identificación de datos atípicos según la cota de 1.5 IQR

```
In [ ]: cota_inferior_1_5 = Q1 - 1.5 * IQR
cota_superior_1_5 = Q3 + 1.5 * IQR
datos_atipicos_1_5 = df[(df < cota_inferior_1_5) | (df > cota_superior_1_5)]
print(f'Número de datos atípicos (1.5 IQR): {len(datos_atipicos_1_5)}')
```

Número de datos atípicos (1.5 IQR): 40

Identificación de datos atípicos según la cota de 3 desviaciones estándar

```
In [ ]: cota_inferior_3_std = media - 3 * std_dev  
cota_superior_3_std = media + 3 * std_dev  
datos_atipicos_3_std = df[(df < cota_inferior_3_std) | (df > cota_superior_3_std)]  
print(f'Número de datos atípicos (3 desviaciones estándar): {len(datos_atipicos_3_std)}
```

Número de datos atípicos (3 desviaciones estándar): 11

Identificación de datos extremos según la cota de 3 IQR

```
In [ ]: cota_inferior_3_IQR = Q1 - 3 * IQR  
cota_superior_3_IQR = Q3 + 3 * IQR  
datos_extremos = df[(df < cota_inferior_3_IQR) | (df > cota_superior_3_IQR)]  
print(f'Número de datos extremos (3 IQR): {len(datos_extremos)}')
```

Número de datos extremos (3 IQR): 13

```
In [ ]: cantidad_1_5_IQR = (len(datos_atipicos_1_5) / len(df)) * 100  
cantidad_1_5_IQR
```

Out[]: 7.259528130671507

Interpreta los datos atípicos

- El análisis con 1.5 IQR identificó varios outliers en el extremo superior de la distribución, lo que sugiere que existen algunos alimentos con un contenido inusualmente alto de grasas monosaturadas. Para ser específicos el 7.25% de los datos son outliers, lo que es considerable al momento de analizar la distribución de la variable.
- En cuanto a los otros outliers de 3 STDEV y 3 IQR, tenemos un menor porcentaje de datos atípicos como es de esperarse.

Análisis de normalidad

Pruebas de normalidad: Anderson-Darling y Jarque Bera

```
In [ ]: ad_test = stats.anderson(df, dist='norm')  
jb_test = stats.jarque_bera(df)
```

```
In [ ]: print(f'Prueba de Anderson-Darling: ')  
ad_test
```

Prueba de Anderson-Darling:

```
Out[ ]: AndersonResult(statistic=46.49930208450678, critical_values=array([0.572, 0.651,  
0.781, 0.911, 1.084]), significance_level=array([15. , 10. , 5. , 2.5, 1. ]))
```

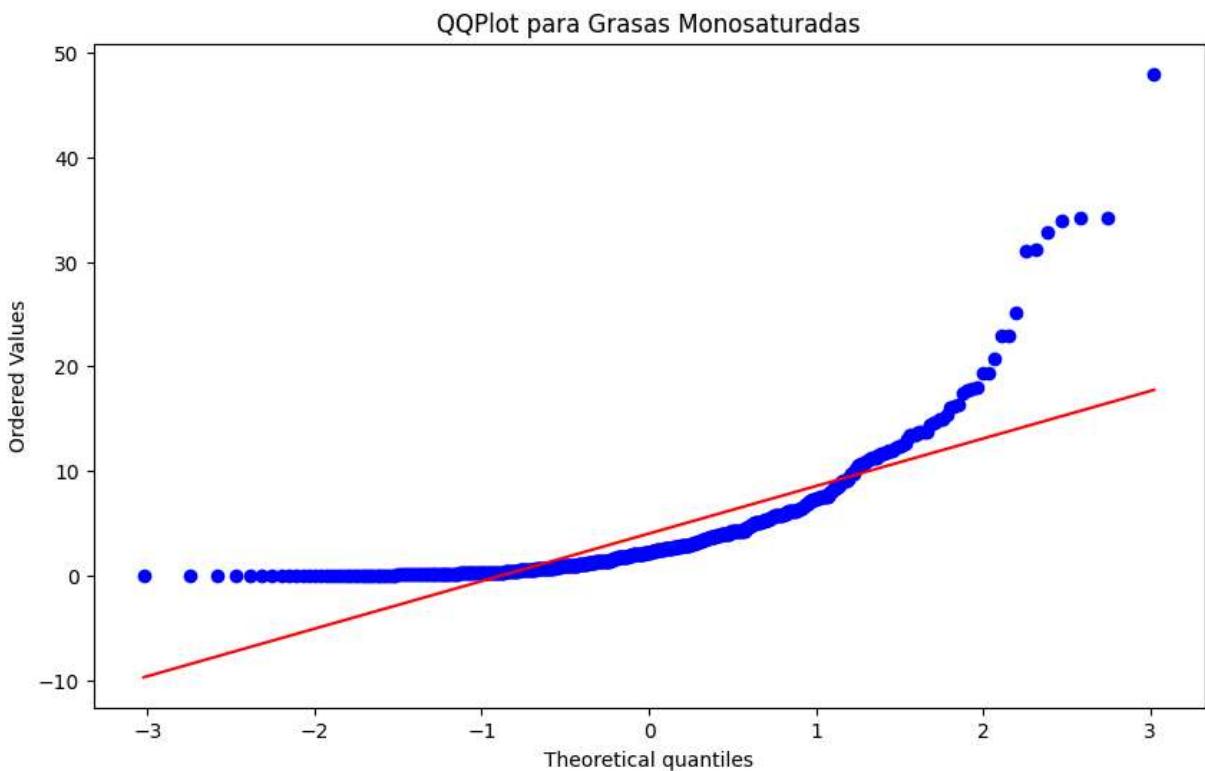
```
In [ ]: print(f'Prueba de Jarque Bera: ')  
jb_test
```

Prueba de Jarque Bera:

```
Out[ ]: Jarque_beraResult(statistic=5883.969556310755, pvalue=0.0)
```

Grafica los datos y su respectivo QQPlot: qqnorm(datos) y qqline(datos)

```
In [ ]: plt.figure(figsize=(10, 6))
stats.probplot(df, dist="norm", plot=plt)
plt.title('QQPlot para Grasas Monosaturadas')
plt.show()
```



Calcula el coeficiente de sesgo y el coeficiente de curtosis

```
In [ ]: sesgo = stats.skew(df)
curtosis = stats.kurtosis(df)
print(f'Coeficiente de sesgo: {sesgo}')
print(f'Coeficiente de curtosis: {curtosis}')
```

Coeficiente de sesgo: 3.207981870795134

Coeficiente de curtosis: 14.66712188368648

Compara las medidas de media, mediana y rango medio de cada variable

```
In [ ]: mediana = np.median(df)
media = np.mean(df)
rango_medio = (Q1 + Q3) / 2

print(f'Media: {media}')
print(f'Mediana: {mediana}')
print(f'Rango Medio: {rango_medio}')
```

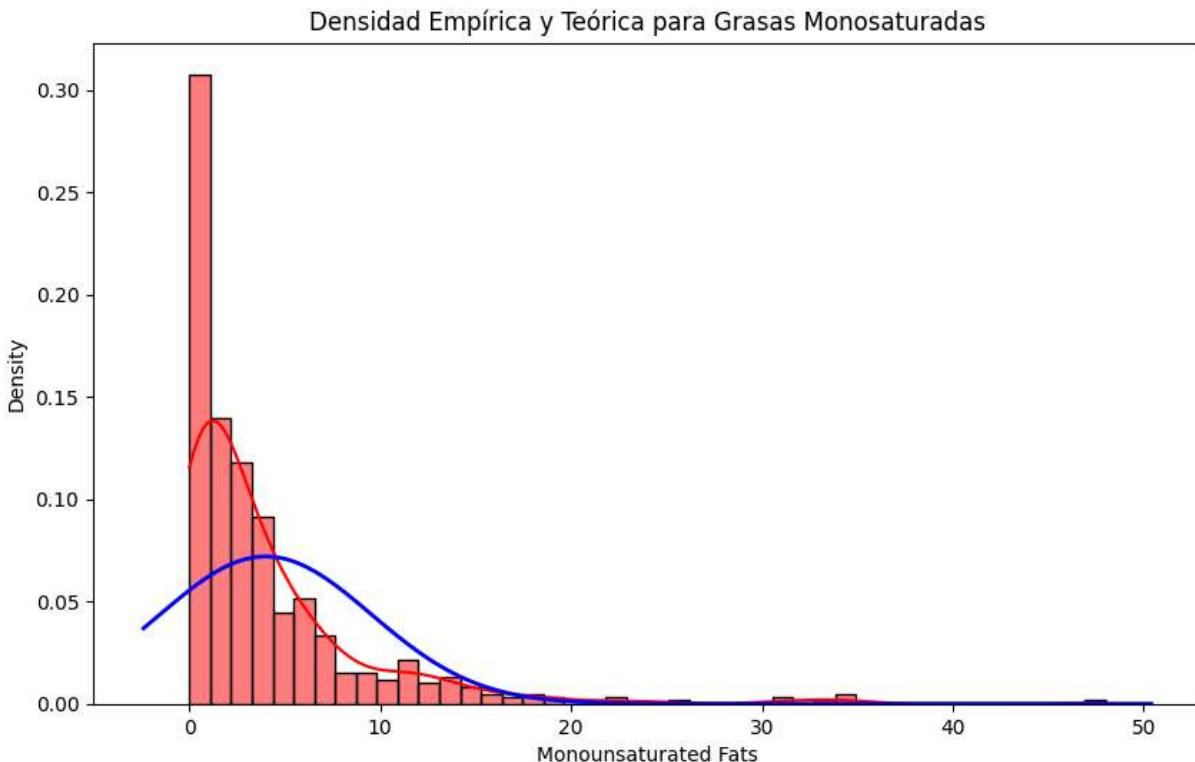
Media: 4.00156442831216

Mediana: 2.2

Rango Medio: 2.875

Realiza el gráfico de densidad empírica y teórica suponiendo normalidad en la variable.

```
In [ ]: plt.figure(figsize=(10, 6))
sns.histplot(df, kde=True, stat="density", color='red')
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
p = stats.norm.pdf(x, media, std_dev)
plt.plot(x, p, 'blue', linewidth=2)
plt.title('Densidad Empírica y Teórica para Grasas Monosaturadas')
plt.show()
```



Interpreta los gráficos y los resultados obtenidos en cada punto con vías a indicar si hay normalidad de los datos y comenta las características encontradas:

- QQplot: Los datos se desvian demasiado de la recta, por lo que no se puede decir que los datos sigan una distribución normal. Por la curva que tiene y que está cerca de los extremos, podemos asumir que la distribución está sesgada.
- Sesgo y curtosis: El sesgo positivo indica que la cola derecha de la distribución es más larga. Tiene una curtosis demasiado alta, por lo que tiene fat tails.
- Media, mediana y rango medio: Podemos decir que la distribución es asimétrica por la diferencia entre la media y la mediana.
- Influencia de los datos atípicos en la normalidad de los datos: Los datos atípicos influyen en la normalidad de los datos, ya que lo podemos ver en los extremos del QQplot.

No hay normalidad en los datos ya que en el test de Jarque Bera el p-valor es menor que 0.05, por lo que los datos no son normales en cuanto a sesgo y curtosis.

Punto 2. Transformación a normalidad

Encuentra la mejor transformación de los datos para lograr normalidad.

```
In [ ]: # Asegura que todos los datos sean positivos  
df_positive = df[df > 0]
```

```
In [ ]: # Transformación Box-Cox  
df_positive_boxcox, lambda_boxcox = boxcox(df_positive)  
  
# Transformación Yeo-Johnson  
df_positive_yeojohnson, lambda_yeojohnson = yeojohnson(df_positive)
```

Escribe las ecuaciones de los modelos de transformación encontrados.

```
In [ ]: # Ecuación de Box-Cox  
equation_boxcox = f'y = (x^{lambda_boxcox}) - 1) / {lambda_boxcox}' if lambda_boxcox != 0 else 'x'  
  
# Ecuación de Yeo-Johnson  
equation_yeojohnson = f'y = ((x + 1)^{lambda_yeojohnson}) - 1) / {lambda_yeojohnson}' if lambda_yeojohnson != 0 else 'x'  
  
print(f'Lambda Box-Cox: {lambda_boxcox}')  
print(f'Ecuación Box-Cox: {equation_boxcox}')  
  
print(f'Lambda Yeo-Johnson: {lambda_yeojohnson}')  
print(f'Ecuación Yeo-Johnson: {equation_yeojohnson}')
```

```
Lambda Box-Cox: 0.19938340746908145  
Ecuación Box-Cox: y = (x^(0.19938340746908145) - 1) / 0.19938340746908145  
Lambda Yeo-Johnson: -0.2458455305343491  
Ecuación Yeo-Johnson: y = ((x + 1)^(-0.2458455305343491) - 1) / -0.2458455305343491
```

```
In [ ]: def summary_stats(data):  
    return {  
        'Mínimo': np.min(data),  
        'Máximo': np.max(data),  
        'Media': np.mean(data),  
        'Mediana': np.median(data),  
        'Cuartil 1': np.percentile(data, 25),  
        'Cuartil 3': np.percentile(data, 75),  
        'Sesgo': stats.skew(data),  
        'Curtosis': stats.kurtosis(data)  
    }  
  
# Calcular estadísticas para datos originales, Box-Cox y Yeo-Johnson  
original_stats = summary_stats(df_positive)  
boxcox_stats = summary_stats(df_positive_boxcox)  
yeojohnson_stats = summary_stats(df_positive_yeojohnson)
```

Compara las medidas: Mínimo, máximo, media, mediana, cuartil 1 y cuartil 3, sesgo y curtosis.

```
In [ ]: print("Estadísticas Originales:")
original_stats
```

Estadísticas Originales:

```
Out[ ]: {'Mínimo': 0.003,
'Máximo': 48.0,
'Media': 4.144477443609023,
'Mediana': 2.3499999999999996,
'Cuartil 1': 0.7,
'Cuartil 3': 5.3,
'Sesgo': 3.1788582086304458,
'Curtosis': 14.37388824268239}
```

```
In [ ]: print("Estadísticas Box-Cox:")
boxcox_stats
```

Estadísticas Box-Cox:

```
Out[ ]: {'Mínimo': -3.4404252484436078,
'Máximo': 5.836857557129127,
'Media': 0.8238743868068105,
'Mediana': 0.93129278353588,
'Cuartil 1': -0.3442878315449959,
'Cuartil 3': 1.9784400705091552,
'Sesgo': -0.044992983486521025,
'Curtosis': -0.28548294178692446}
```

```
In [ ]: print("Estadísticas Yeo-Johnson:")
yeojohnson_stats
```

Estadísticas Yeo-Johnson:

```
Out[ ]: {'Mínimo': 0.0029944062554342634,
'Máximo': 2.5051289550172093,
'Media': 1.0138232164205478,
'Mediana': 1.045747030072531,
'Cuartil 1': 0.4974745351270056,
'Cuartil 3': 1.4804342376332655,
'Sesgo': 0.06290888691219346,
'Curtosis': -0.9314623518392127}
```

Grafica las funciones de densidad empírica y teórica de los 2 modelos obtenidos (exacto y aproximado) y los datos originales.

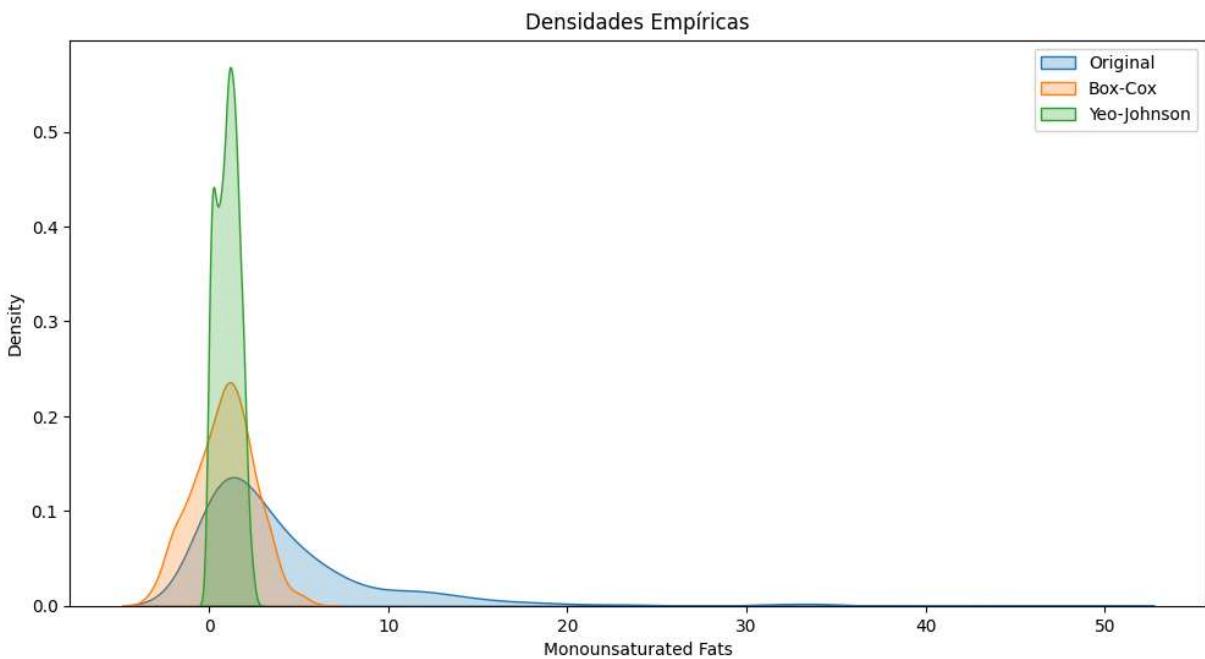
```
In [ ]: # Función para graficar densidades
def plot_density(data, label):
    sns.kdeplot(data, label=label, fill=True)

# Graficar densidades
plt.figure(figsize=(12, 6))
plot_density(df_positive, 'Original')
plot_density(df_positive_boxcox, 'Box-Cox')
plot_density(df_positive_yeojohnson, 'Yeo-Johnson')
```

```

plt.title('Densidades Empíricas')
plt.legend()
plt.show()

```



Realiza la prueba de normalidad de Anderson-Darling y de Jarque Bera para los datos transformados y los originales

```

In [ ]: # Prueba de Anderson-Darling
def ad_test(data, label):
    statistic, p_value = normal_ad(data)
    print(f'Anderson-Darling para {label}: Estadístico = {statistic:.3f}, p-valor = {p_value:.3f}')

# Realizar pruebas de normalidad
ad_test(df_positive, 'Original')
ad_test(df_positive_boxcox, 'Box-Cox')
ad_test(df_positive_yeojohnson, 'Yeo-Johnson')

```

Anderson-Darling para Original: Estadístico = 44.108, p-valor = 0.000

Anderson-Darling para Box-Cox: Estadístico = 0.807, p-valor = 0.037

Anderson-Darling para Yeo-Johnson: Estadístico = 3.630, p-valor = 0.000

```

In [ ]: # Prueba de Jarque-Bera
def jb_test(data, label):
    statistic, p_value = stats.jarque_bera(data)
    print(f'Jarque-Bera para {label}: Estadístico = {statistic:.3f}, p-valor = {p_value:.3f}')

# Realizar pruebas de Jarque-Bera
jb_test(df_positive, 'Original')
jb_test(df_positive_boxcox, 'Box-Cox')
jb_test(df_positive_yeojohnson, 'Yeo-Johnson')

```

Jarque-Bera para Original: Estadístico = 5475.814, p-valor = 0.000

Jarque-Bera para Box-Cox: Estadístico = 1.986, p-valor = 0.370

Jarque-Bera para Yeo-Johnson: Estadístico = 19.583, p-valor = 0.000

Detecta anomalías y corrige tu base de datos (datos atípicos, ceros anámalos, etc).

```
In [ ]: # Identificar y eliminar outliers usando IQR
Q1 = np.percentile(df_positive, 25)
Q3 = np.percentile(df_positive, 75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Filtrar datos sin outliers
filtered_data = df_positive[(df_positive >= lower_bound) & (df_positive <= upper_bound)]
```

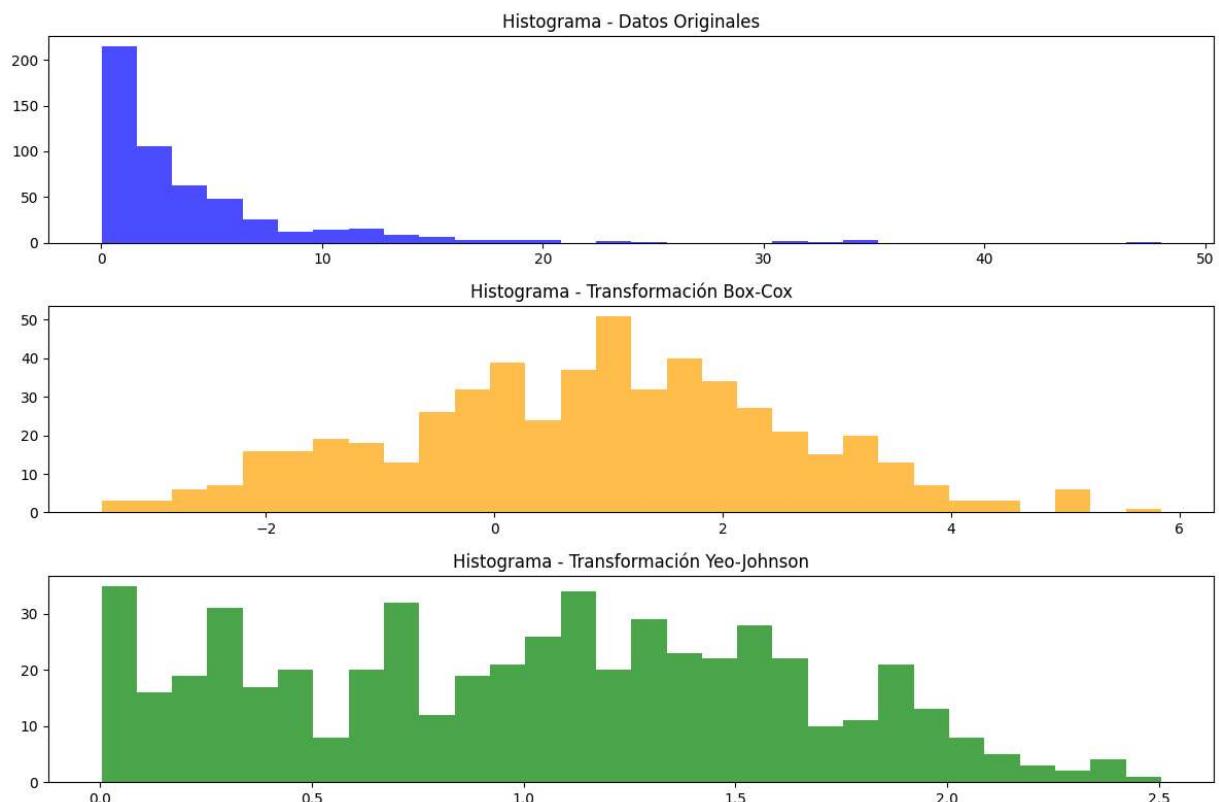
Histograma de los datos originales y transformados

```
In [ ]: plt.figure(figsize=(12, 8))
plt.subplot(3, 1, 1)
plt.hist(df_positive, bins=30, color='blue', alpha=0.7)
plt.title('Histograma - Datos Originales')

plt.subplot(3, 1, 2)
plt.hist(df_positive_boxcox, bins=30, color='orange', alpha=0.7)
plt.title('Histograma - Transformación Box-Cox')

plt.subplot(3, 1, 3)
plt.hist(df_positive_yeojohnson, bins=30, color='green', alpha=0.7)
plt.title('Histograma - Transformación Yeo-Johnson')

plt.tight_layout()
plt.show()
```



Comenta la normalidad de las transformaciones obtenidas. Utiliza como argumento de normalidad:

- En cuanto a las medidas (mínimo, máximo, media, mediana, cuartil 1, etc.) Yeo-Johnson se comporta decente en cuanto a la media y mediana, pero Box-Cox es mejor en cuanto a las demás medidas. Por lo que Box-Cox se adapta mejor a los datos.
- A simple vista, viendo los histogramas, el de Box-Cox se acerca más a una distribución normal.
- De acuerdo a la prueba de normalidad de Anderson-Darling y Jarque Bera para los datos transformados y los originales la única distribución que no es rechazada rotundamente como normal es la de Box-Cox.

Los principales motivos de alejamiento de normalidad que veo son:

- Los datos atípicos ya que se pueden ver claramente en los extremos del QQplot.
- La alta curtosis que indica fat tails en la distribución original.
- El sesgo alto que indica asimetría en la distribución original.

En conclusión, la mejor transformación de los datos de acuerdo a las características de los modelos que encontré es Box-Cox.