actividadintegradoraprobafer

August 20, 2024

###Actividad integradora 1 ##Fernanda Pérez ##A01742102

##Punto 1. Análisis descriptivo de la variable

Analiza una de las siguientes variables en cuanto a sus datos atípicos y normalidad. La variable que te corresponde analizar te será asignada por tu profesora al inicio de la actividad: 2.Grasas saturadas

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import anderson, jarque_bera
import matplotlib.pyplot as plt
import scipy.stats as stats
import numpy as np
from sklearn.preprocessing import PowerTransformer
from scipy.stats import skew, kurtosis
import seaborn as sns
```

[2]:
```
file_path = '/content/food_data_g.csv'
df = pd.read_csv(file_path)

df.head()
```

[2]:

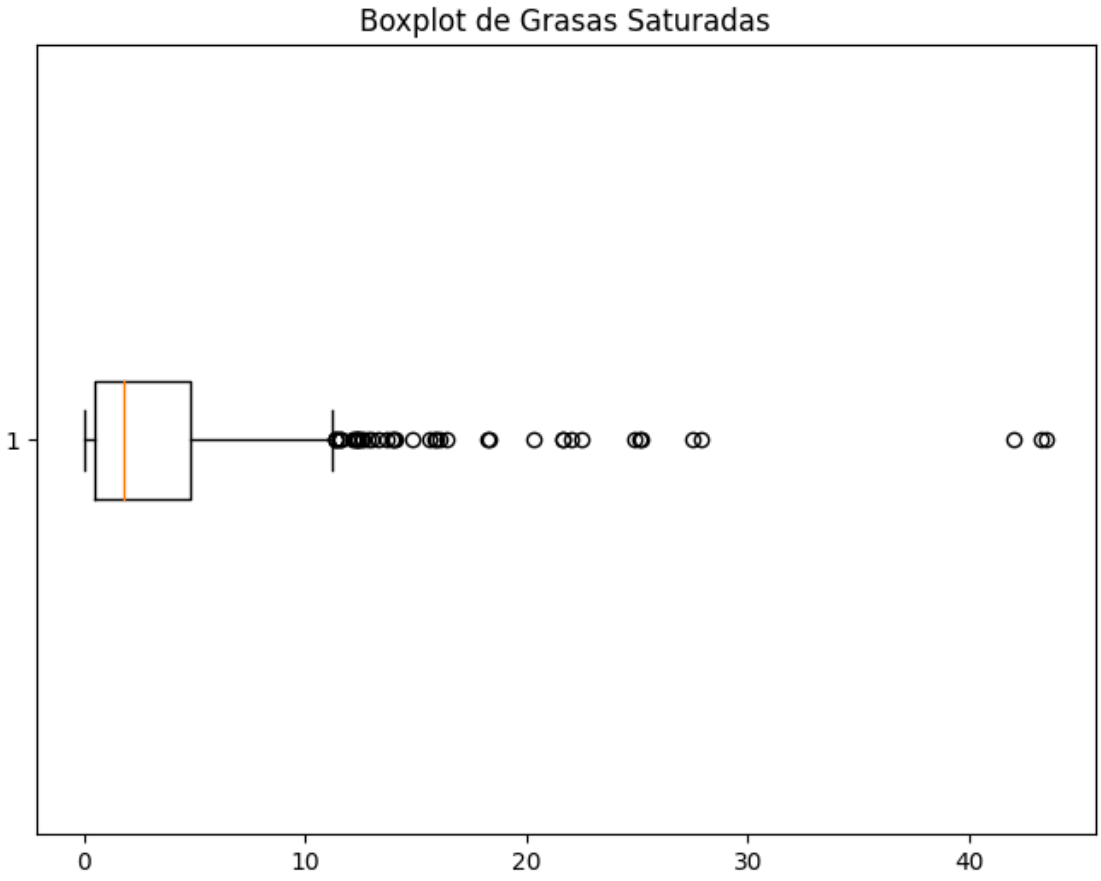
	Unnamed: 0.1	Unnamed: 0	food	Caloric Value	Fat	Saturated Fats	Monounsaturated Fats	Polyunsaturated Fats	Carbohydrates	Sugars	…	Calcium	Copper	Iron	Magnesium	Manganese
0	0	0	cream cheese	51	5.0	2.9	1.3	0.200	0.8	0.500	…	0.008	14.100	0.082	0.027	1.300
1	1	1	neufchatel cheese	215	19.4	10.9	4.9	0.800								
2	2	2	requeijao cremoso light catupiry	49	3.6	2.3	0.9	0.000								
3	3	3	ricotta cheese	30	2.0	1.3	0.5	0.002								
4	4	4	cream cheese low fat	30	2.3	1.4	0.6	0.042								

```
1            3.1    2.700  …    99.500    0.034   0.100       8.500       0.088
2            0.9    3.400  …     0.000    0.000   0.000       0.000       0.000
3            1.5    0.091  …     0.097   41.200   0.097       0.096       4.000
4            1.2    0.900  …    22.200    0.072   0.008       1.200       0.098

   Phosphorus  Potassium  Selenium   Zinc  Nutrition Density
0       0.091       15.5    19.100  0.039              7.070
1     117.300      129.2     0.054  0.700            130.100
2       0.000        0.0     0.000  0.000              5.400
3       0.024       30.8    43.800  0.035              5.196
4      22.800       37.1     0.034  0.053             27.007

[5 rows x 37 columns]
```

1. Para analizar datos atípicos se te sugiere:

Graficar el diagrama de caja y bigote

[3]:
```
plt.figure(figsize=(8, 6))
plt.boxplot(df['Saturated Fats'].dropna(), vert=False)
plt.title('Boxplot de Grasas Saturadas')
plt.show()
```



2. Calcula las principales medidas que te ayuden a identificar datos atípicos (utilizar summary te puede abreviar el cálculo): Cuartil 1, Cuartil 3, Media, Cuartil 3, Rango intercuartílico y Desviación estándar

[4]:
```
Q1 = df['Saturated Fats'].quantile(0.25)
Q2 = df['Saturated Fats'].median()
Q3 = df['Saturated Fats'].quantile(0.75)
IQR = Q3 - Q1
std_dev = df['Saturated Fats'].std()

print(f"Cuartil 1 (Q1): {Q1}")
print(f"Mediana (Q2): {Q2}")
print(f"Cuartil 3 (Q3): {Q3}")
print(f"Rango Intercuartílico (IQR): {IQR}")
print(f"Desviación Estándar: {std_dev}")
```

```
Cuartil 1 (Q1): 0.5
Mediana (Q2): 1.8
Cuartil 3 (Q3): 4.8
Rango Intercuartílico (IQR): 4.3
Desviación Estándar: 5.397020953732459
```

Identifica la cota de 1.5 rangos intercuartílicos para datos atípicos, ¿hay datos atípicos de acuerdo con este criterio? ¿cuántos son?

[5]:
```
lower_bound_1_5_iqr = Q1 - 1.5 * IQR
upper_bound_1_5_iqr = Q3 + 1.5 * IQR

outliers_1_5_iqr = df[(df['Saturated Fats'] < lower_bound_1_5_iqr) |␣
 ↪(df['Saturated Fats'] > upper_bound_1_5_iqr)]
outliers_count_1_5_iqr = outliers_1_5_iqr.shape[0]

print(f"Se tienen datos atípicos de acuerdo al criterio:␣
 ↪{outliers_count_1_5_iqr}")
```

```
Se tienen datos atípicos de acuerdo al criterio: 42
```

Identifica la cota de 3 desviaciones estándar alrededor de la media, ¿hay datos atípicos de acuerdo con este criterio? ¿cuántos son?

[6]:
```
mean = df['Saturated Fats'].mean()
lower_bound_3_std = mean - 3 * std_dev
upper_bound_3_std = mean + 3 * std_dev

outliers_3_std = df[(df['Saturated Fats'] < lower_bound_3_std) | (df['Saturated␣
 ↪Fats'] > upper_bound_3_std)]
outliers_count_3_std = outliers_3_std.shape[0]
```

```
print(f"De acuerdo a este criterio hay datos atípicos: {outliers_count_3_std}")
```

```
De acuerdo a este criterio hay datos atípicos: 13
```

Identifica la cota de 3 rangos intercuartílicos para datos extremos, ¿hay datos extremos de acuerdo con este criterio? ¿cuántos son?

[7]:
```
lower_bound_3_iqr = Q1 - 3 * IQR
upper_bound_3_iqr = Q3 + 3 * IQR

extreme_outliers_3_iqr = df[(df['Saturated Fats'] < lower_bound_3_iqr) |␣
 ↪(df['Saturated Fats'] > upper_bound_3_iqr)]
extreme_outliers_count_3_iqr = extreme_outliers_3_iqr.shape[0]

print(f"De acuerdo a este criterio hay datos extremos R:␣
 ↪{extreme_outliers_count_3_iqr}")
```

```
De acuerdo a este criterio hay datos extremos R: 15
```

Interpreta los resultados obtenidos y argumenta sobre el comportamiento de los datos atípicos y extremos en la variable seleccionada

Al analizar mi varible 2.Saturated Fats se pueden identificar varias datos atípicos y extemos los cuales influyen en el momento de interpretar los resultados. Al utilizar la cota de 1.5 IQR encontré 42 datos atípicos, o sea que hay una cantidad considerable de valores que se salen del rango esperado. Y al revisar las 3 desviaciones estandar se observa que hay 13 datos atípicos extremos, o sea que hay varios valores que se alejan de la media. Al tener estos resultados se sugiere tratarlos ya que podría haber distorsiones en el anáñisis.

##2. Para analizar normalidad se te sugiere:

Realiza pruebas de normalidad univariada para la variable (utiliza las pruebas de Anderson-Darling y de Jarque Bera). No olvides incluir H0 y H1 para la prueba de normalidad.

Prueba Anderson-Darling:

[8]:
```
result_ad = anderson(df['Saturated Fats'].dropna())
print(f"Statistic: {result_ad.statistic}")
for i in range(len(result_ad.critical_values)):
    sl, cv = result_ad.significance_level[i], result_ad.critical_values[i]
    print(f"Significance Level {sl}: Critical Value {cv}")
```

```
Statistic: 50.09371521015373
Significance Level 15.0: Critical Value 0.572
Significance Level 10.0: Critical Value 0.651
Significance Level 5.0: Critical Value 0.781
Significance Level 2.5: Critical Value 0.911
Significance Level 1.0: Critical Value 1.084
```

Prueba Jarque Bera

[9]:
```
stat, p_value = jarque_bera(df['Saturated Fats'].dropna())
print(f"Statistic: {stat}, p-value: {p_value}")
```

```
Statistic: 7694.104934424058, p-value: 0.0
```

H0 y H1

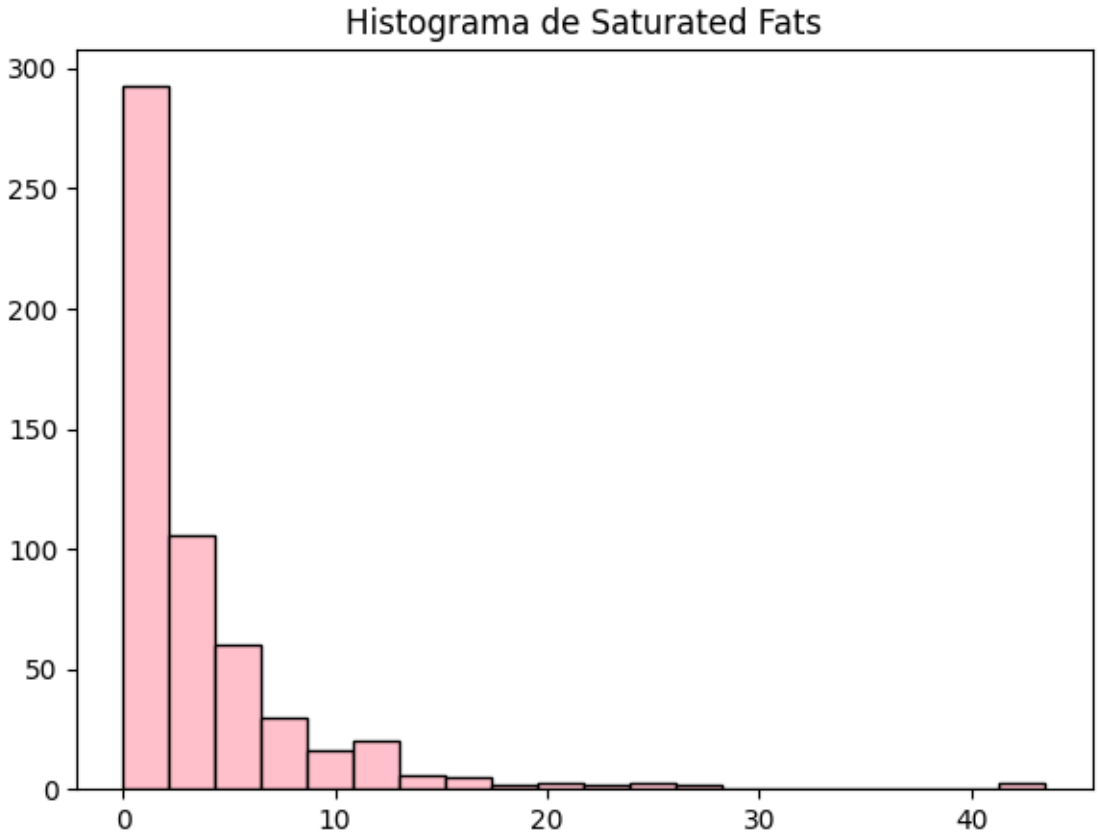
H0: Los datos siguen una distribución normal. H1: Los datos no siguen una distribución normal.

Por los resultados que se acaban de obtener me doy cuenta que en ambas se rechaza la hipótesis nula (H0) y se acepta (H1). O sea que mi variable 'Saturated Fats' no sigue una distribución normal.

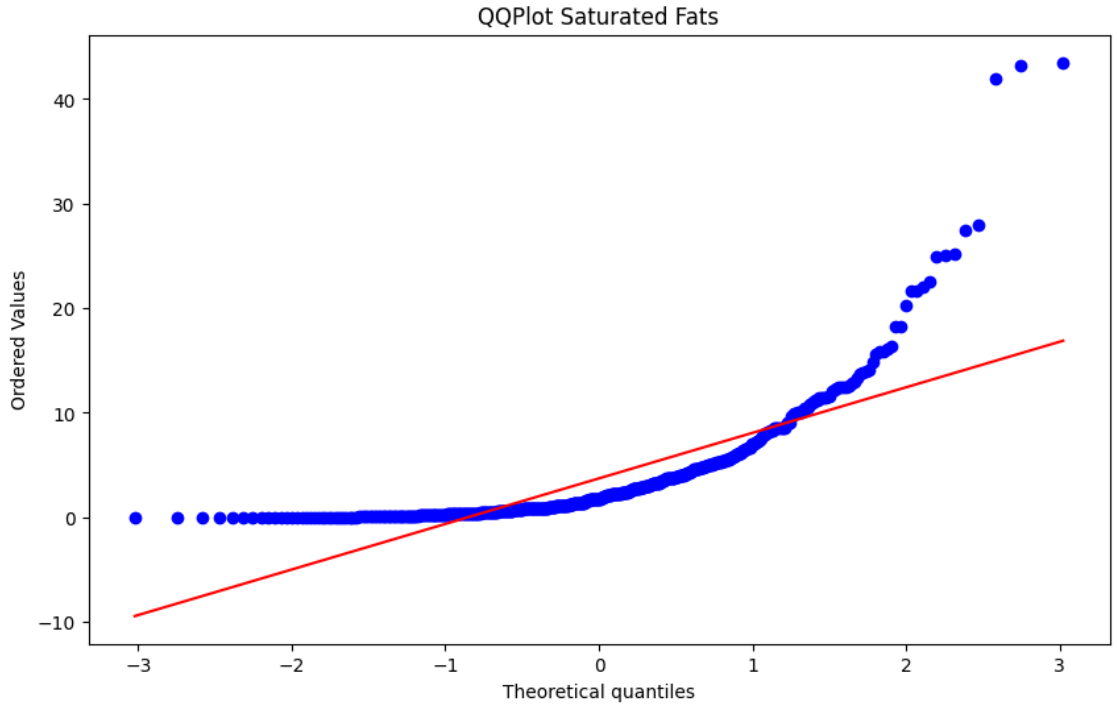
Grafica los datos y su respectivo QQPlot: qqnorm(datos) y qqline(datos)

[10]:
```
plt.figure(figsize=(15, 5))
plt.subplot(1, 2, 1)
plt.hist(df['Saturated Fats'].dropna(), bins=20, color='pink',␣
 ↪edgecolor='black')
plt.title('Histograma de Saturated Fats')
```

[10]:
```
Text(0.5, 1.0, 'Histograma de Saturated Fats')
```



[11]:
```
plt.figure(figsize=(10, 6))
stats.probplot(df['Saturated Fats'], dist="norm", plot=plt)
plt.title('QQPlot Saturated Fats')
plt.show()
```



Calcula el coeficiente de sesgo y el coeficiente de curtosis

[12]:
```
skewness = df['Saturated Fats'].skew()
print(f"Coeficiente de sesgo : {skewness}")

kurtosis = df['Saturated Fats'].kurtosis()
print(f"Coeficiente de curtosis: {kurtosis}")
```

```
Coeficiente de sesgo : 3.437997319625104
Coeficiente de curtosis: 17.13985318084311
```

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

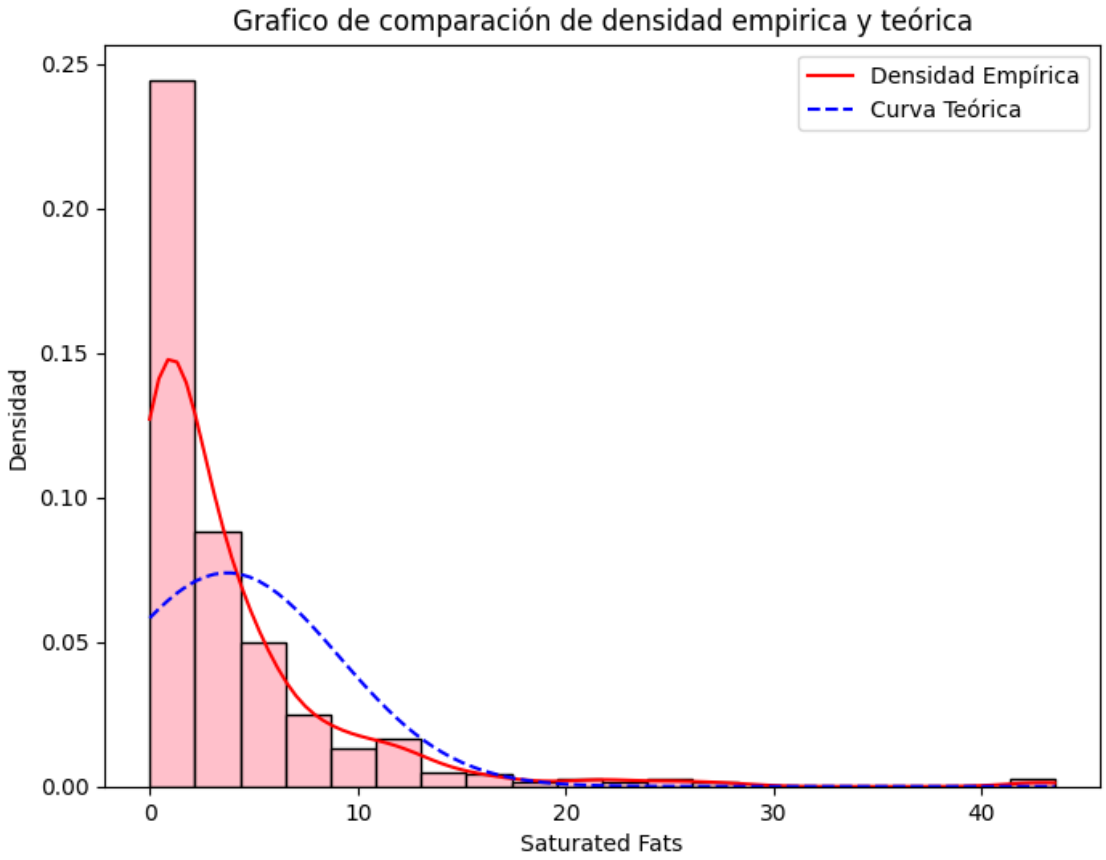
[13]:
```
mean_value = df['Saturated Fats'].mean()
print(f"Media: {mean_value}")

median_value = df['Saturated Fats'].median()
print(f"Mediana: {median_value}")

range_value = df['Saturated Fats'].max() - df['Saturated Fats'].min()
```

```
print(f"Rango Medio: {range_value / 2}")
```

```
Media: 3.7227150635208717
Mediana: 1.8
Rango Medio: 21.75
```

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

[14]:
```
plt.figure(figsize=(8, 6))
plt.hist(df['Saturated Fats'].dropna(), bins=20, density=True, color='pink',␣
 ↪edgecolor='black')

density = stats.gaussian_kde(df['Saturated Fats'].dropna())
x = np.linspace(min(df['Saturated Fats']), max(df['Saturated Fats']), 100)
plt.plot(x, density(x), color='red', label='Densidad Empírica')

mean_value = df['Saturated Fats'].mean()
std_dev_value = df['Saturated Fats'].std()
plt.plot(x, stats.norm.pdf(x, mean_value, std_dev_value), color='blue',␣
 ↪linestyle='dashed', label='Curva Teórica')

plt.title('Grafico de comparación de densidad empirica y teórica')
plt.xlabel('Saturated Fats')
plt.ylabel('Densidad')
plt.legend()
plt.show()
```



Interpreta los gráficos y los resultados obtenidos en cada punto con vías a indicar si hay normalidad de los datos. Comenta las características encontradas: Considera alejamientos de normalidad por simetría, curtosis Comenta si hay aparente influencia de los datos atípicos en la normalidad de los datos Emite una conclusión sobre la normalidad de los datos. Se debe argumentar en términos de los 3 puntos analizados: las pruebas de normalidad, los gráficos y las medidas:

Los resultados de mi prueba de normalidad de Anderson Darling y Jarque Bera me indican que mi variable no sigue una distribucion normal porque en las 2 pruebas se rechazó la hipotesis nuela de la normalida.En el histograma vemos una distribución sesgada a la derecha lo cual tambien vemos reflejado en el qq plot, ya que vemos como los datos se desvian de la linea de normalidad esperada, en espacial lo vemos en los extremos. Como obtuve el coeficiente de sesgo positivo '3.4379' y una curtosis alta '17.1398' hay asimestria.

Y en la La comparación entre la densidad empírica y la curva teórica normal vemos que en los valores bajos es donde se concentran más datos y tiene cola hacia los valores altos, o sea no es distribución normal. Y como la media es mayor que la mediana '3.7227 > 1.8' es una distribución sesgada.

Dado que no sigue una distribución normal debido a la asimetría, la presencia de valores atípicos y la alta curtosis es que se llega a la conclusión de que los dtaos de mi variable 'Saturated Fats' no presenta normalidad.

##Punto 2. Transformación a normalidad

Encuentra la mejor transformación de los datos para lograr normalidad. Puedes hacer uso de la transformación Box-Cox o de Yeo Johnson o el comando powerTransform para encontrar la mejor lambda para la transformación. Utiliza el modelo exacto y el aproximado de acuerdo con las sugerencias de Box y Cox para la transformación.

[15]:
```
df_filtered = df[df['Saturated Fats'] > 0].dropna()

box_cox_transformer = PowerTransformer(method='box-cox', standardize=False)
saturated_fats_boxcox = box_cox_transformer.
 ↪fit_transform(df_filtered[['Saturated Fats']])

yeo_johnson_transformer = PowerTransformer(method='yeo-johnson',␣
 ↪standardize=False)
saturated_fats_yeojohnson = yeo_johnson_transformer.
 ↪fit_transform(df[['Saturated Fats']].dropna())

print(f"Lambda de Box-Cox: {box_cox_transformer.lambdas_}")
print(f"Lambda de Yeo-Johnson: {yeo_johnson_transformer.lambdas_}")
```

```
Lambda de Box-Cox: [0.18701272]
Lambda de Yeo-Johnson: [-0.3364272]
```

Escribe las ecuaciones de los modelos de transformación encontrados.

[16]:
```
lambda_boxcox = box_cox_transformer.lambdas_[0]
lambda_yeojohnson = yeo_johnson_transformer.lambdas_[0]

equation_boxcox = f'y = (x^({lambda_boxcox:.6f}) - 1) / {lambda_boxcox:.6f}' if␣
 ↪lambda_boxcox != 0 else 'y = ln(x)'

equation_yeojohnson = f'y = ((x + 1)^({lambda_yeojohnson:.6f}) - 1) /␣
 ↪{lambda_yeojohnson:.6f}' if lambda_yeojohnson != 0 else 'y = ln(x + 1)'

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: 0.1870127155082245
Ecuación Box-Cox: y = (x^(0.187013) - 1) / 0.187013
Lambda Yeo-Johnson: -0.33642720380541674
Ecuación Yeo-Johnson: y = ((x + 1)^(-0.336427) - 1) / -0.336427
```

Analiza la normalidad de las transformaciones obtenidas con los datos originales. Utiliza como argumento de normalidad:

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

[20]:
```
from scipy.stats import skew, kurtosis

original_stats = {
    "min": df['Saturated Fats'].min(),
    "max": df['Saturated Fats'].max(),
    "mean": df['Saturated Fats'].mean(),
    "median": df['Saturated Fats'].median(),
    "Q1": df['Saturated Fats'].quantile(0.25),
    "Q3": df['Saturated Fats'].quantile(0.75),
    "skewness": skew(df['Saturated Fats'].dropna()),
    "kurtosis": kurtosis(df['Saturated Fats'].dropna())
}

original_stats_df = pd.DataFrame([original_stats], index=["Original"])
print("Medidas para los datos originales:")
print(original_stats_df)
print("\n")
```

```
Medidas para los datos originales:
             min   max      mean  median   Q1   Q3  skewness   kurtosis
Original  0.0  43.5  3.722715     1.8  0.5  4.8  3.428631  16.973844
```

[21]:
```
boxcox_stats = {
    "min": saturated_fats_boxcox.min(),
    "max": saturated_fats_boxcox.max(),
    "mean": saturated_fats_boxcox.mean(),
    "median": np.median(saturated_fats_boxcox),
    "Q1": np.percentile(saturated_fats_boxcox, 25),
    "Q3": np.percentile(saturated_fats_boxcox, 75),
    "skewness": skew(saturated_fats_boxcox),
    "kurtosis": kurtosis(saturated_fats_boxcox)
}

boxcox_stats_df = pd.DataFrame([boxcox_stats], index=["Box-Cox"])
print("Medidas para los datos transformados con Box-Cox:")
print(boxcox_stats_df)
print("\n")
```

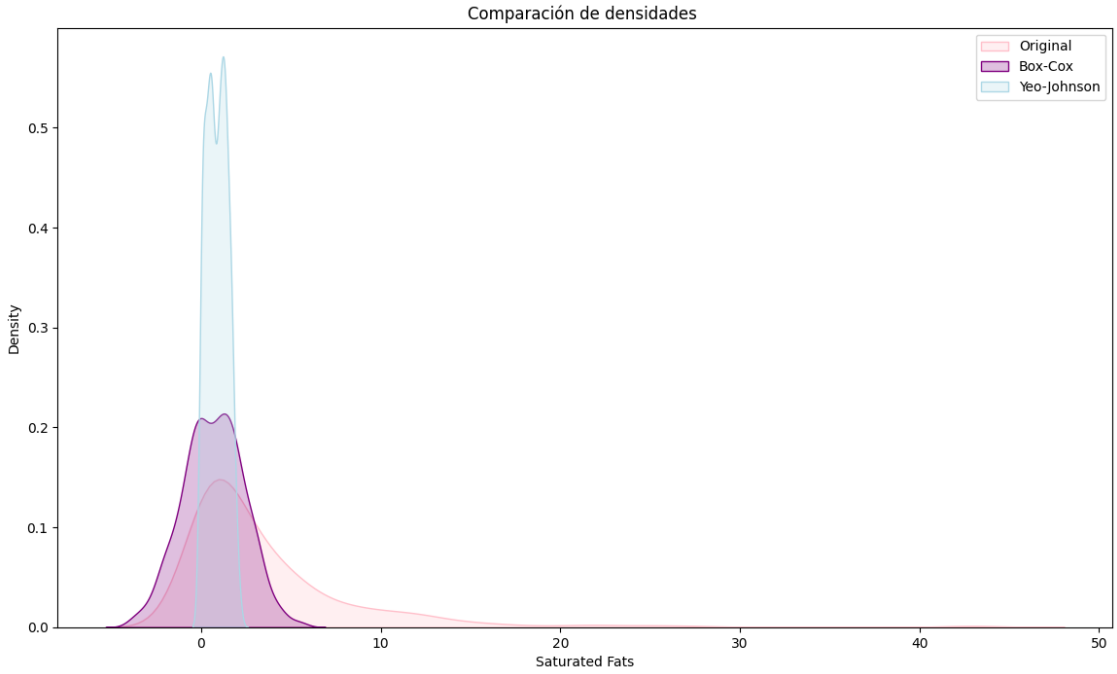
```
Medidas para los datos transformados con Box-Cox:
              min       max      mean    median        Q1        Q3  \
Box-Cox -3.877997  5.480744  0.673859  0.740078 -0.487185  1.850645

                        skewness                kurtosis
Box-Cox  [-0.02852522260035111]  [-0.20751985608837042]
```

[22]:
```
yeojohnson_stats = {
    "min": saturated_fats_yeojohnson.min(),
    "max": saturated_fats_yeojohnson.max(),
    "mean": saturated_fats_yeojohnson.mean(),
    "median": np.median(saturated_fats_yeojohnson),
    "Q1": np.percentile(saturated_fats_yeojohnson, 25),
    "Q3": np.percentile(saturated_fats_yeojohnson, 75),
    "skewness": skew(saturated_fats_yeojohnson),
    "kurtosis": kurtosis(saturated_fats_yeojohnson)
}

yeojohnson_stats_df = pd.DataFrame([yeojohnson_stats], index=["Yeo-Johnson"])
print("Medidas para los datos transformados con Yeo-Johnson:")
print(yeojohnson_stats_df)
```

```
Medidas para los datos transformados con Yeo-Johnson:
              min       max      mean    median        Q1        Q3  \
Yeo-Johnson -0.0  2.143409  0.871509  0.870217  0.379026  1.327013

                           skewness               kurtosis
Yeo-Johnson  [0.10220428055207641]  [-1.0721332861182984]
```

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

[23]:
```
plt.figure(figsize=(14, 8))

sns.kdeplot(df['Saturated Fats'].dropna(), label="Original", color='pink',␣
 ↪fill=True)

sns.kdeplot(saturated_fats_boxcox.flatten(), label="Box-Cox", color='purple',␣
 ↪fill=True)

sns.kdeplot(saturated_fats_yeojohnson.flatten(), label="Yeo-Johnson",␣
 ↪color='lightblue', fill=True)

plt.title("Comparación de densidades ")
plt.legend()
plt.show()
```



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

[24]:
```
ad_original = anderson(df['Saturated Fats'].dropna())
jb_original = jarque_bera(df['Saturated Fats'].dropna())

ad_boxcox = anderson(saturated_fats_boxcox.flatten())
jb_boxcox = jarque_bera(saturated_fats_boxcox.flatten())

ad_yeojohnson = anderson(saturated_fats_yeojohnson.flatten())
jb_yeojohnson = jarque_bera(saturated_fats_yeojohnson.flatten())

print(f"Original - Anderson-Darling: {ad_original.statistic}, Jarque-Bera:␣
  ↪{jb_original}")
print(f"Box-Cox - Anderson-Darling: {ad_boxcox.statistic}, Jarque-Bera:␣
  ↪{jb_boxcox}")
print(f"Yeo-Johnson - Anderson-Darling: {ad_yeojohnson.statistic}, Jarque-Bera:␣
  ↪{jb_yeojohnson}")
```

```
Original - Anderson-Darling: 50.09371521015373, Jarque-Bera:
SignificanceResult(statistic=7694.104934424058, pvalue=0.0)
Box-Cox - Anderson-Darling: 0.46786560606642524, Jarque-Bera:
SignificanceResult(statistic=1.0306031782999325, pvalue=0.5973204220922074)
Yeo-Johnson - Anderson-Darling: 5.442644672039705, Jarque-Bera:
SignificanceResult(statistic=27.34917526348232, pvalue=1.1513359067581793e-06)
```

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

[25]:
```
Q1 = df['Saturated Fats'].quantile(0.25)
Q3 = df['Saturated Fats'].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

df_clean = df[(df['Saturated Fats'] >= lower_bound) & (df['Saturated Fats'] <=␣
 ↪upper_bound)]

print(f"Tamaño original: {df.shape[0]}")
print(f"Tamaño después de eliminar atípicos: {df_clean.shape[0]}")
```

```
Tamaño original: 551
Tamaño después de eliminar atípicos: 509
```

Como se ve se eliminan los 42 datos que obtuve en este inciso “Identifica la cota de 1.5 rangos intercuartílicos para datos atípicos, ¿hay datos atípicos de acuerdo con este criterio? ¿cuántos son?”

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

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

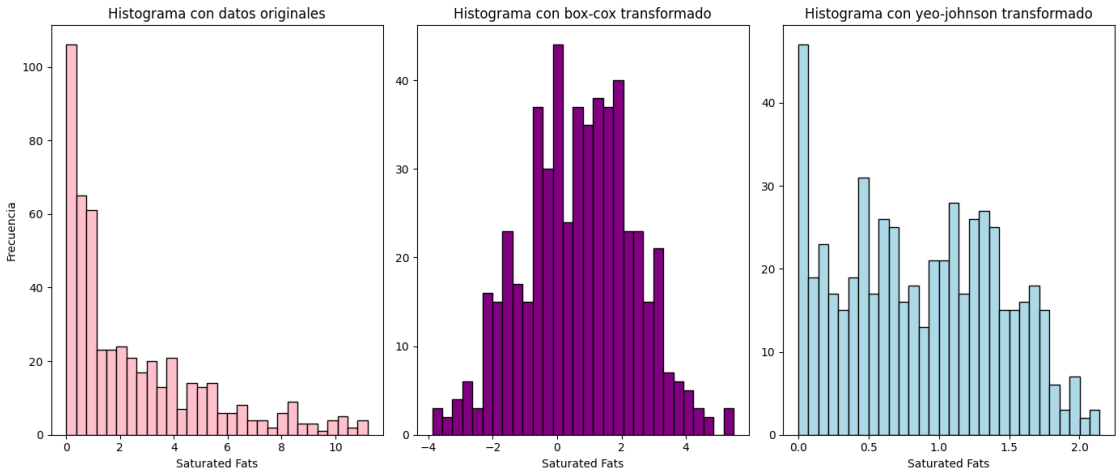
[26]:
```
print("Medidas para los datos originales después de limpiar:")
print(original_stats_df)
print("\nMedidas para los datos transformados con Box-Cox después de limpiar:")
print(boxcox_stats_df)
print("\nMedidas para los datos transformados con Yeo-Johnson después de␣
 ↪limpiar:")
print(yeojohnson_stats_df)
```

```
Medidas para los datos originales después de limpiar:
              min   max      mean  median   Q1   Q3  skewness   kurtosis
Original  0.0  43.5  3.722715     1.8  0.5  4.8  3.428631  16.973844

Medidas para los datos transformados con Box-Cox después de limpiar:
                min       max      mean    median        Q1        Q3  \
Box-Cox -3.877997  5.480744  0.673859  0.740078 -0.487185  1.850645

                          skewness                kurtosis
Box-Cox  [-0.02852522260035111]  [-0.20751985608837042]

Medidas para los datos transformados con Yeo-Johnson después de limpiar:
                min       max      mean    median        Q1        Q3  \
Yeo-Johnson -0.0  2.143409  0.871509  0.870217  0.379026  1.327013
```

```
                          skewness                kurtosis
Yeo-Johnson   [0.10220428055207641]   [-1.0721332861182984]
```

Obten el histograma de los 2 modelos obtenidos (exacto y aproximado) y de los datos originales.

[27]:
```
plt.figure(figsize=(14, 6))
plt.subplot(1, 3, 1)
plt.hist(df_clean['Saturated Fats'], bins=30, color='pink', edgecolor='black')
plt.title('Histograma con datos originales')
plt.xlabel('Saturated Fats')
plt.ylabel('Frecuencia')


plt.subplot(1, 3, 2)
plt.hist(saturated_fats_boxcox, bins=30, color='purple', edgecolor='black')
plt.title('Histograma con box-cox transformado')
plt.xlabel('Saturated Fats')


plt.subplot(1, 3, 3)
plt.hist(saturated_fats_yeojohnson, bins=30, color='lightblue',␣
 ↪edgecolor='black')
plt.title('Histograma con yeo-johnson transformado')
plt.xlabel('Saturated Fats')

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



Indica posibilidades de motivos de alejamiento de normalidad (sesgo, curtosis, datos atípicos, etc)

Sesgo: Como los datos originales presentan un sesgo significativo, nos indica una asimetría en la distribución de los datos. Curtosis: Al tener una curtosis elevada en los datos originales vemos que

hay picos altos y colas pesadas, por lo que la distribución se aleja de la normalidad. Datos atípicos: Como existen varios valores atípicos en los datos originales también afecta la normalidad, haciendo que la distribución se desvíe aún más de la normal.

Define la mejor transformación de los datos de acuerdo a las características de los modelos que encontraste. Toma en cuenta los criterios del inciso anterior para analizar normalidad y la economía del modelo.

Basandome en las pruebas de normalidad, el análisis del sesgo y la curtosis, y en la información visual que obtuve de los histogramas, la transformación Yeo-Johnson es la más efectiva para normalizar mi variable 'saturated fats'ya que obtengo mejores resultados con esta.