ANÁLISE DE OUTLIER

Thiago Seiki Kato - 17/07/2024

O QUE É UM OUTLIER?

Um outlier, também conhecido como valor atípico, ponto fora da curva ou dado discrepante, é uma observação em um conjunto de dados que se desvia significativamente dos demais valores. Essa discrepância pode ser causada por diversos fatores, como erros de medição, eventos raros ou características únicas daquela observação.

	Dados sem outlier	Dados com outlier
Dados	1,2,3,3,4,5,4	1,2,3,3,4,5,**400**
Média	3.14	**59.714**
**Mediana*	3	3
Desvio Padrão	1.345185	**150.057**

Como você pode ver, o conjunto de dados com valores discrepantes tem média e desvio padrão significativamente diferentes. No primeiro cenário, diremos que a média é 3,14. Mas com o valor discrepante, a média sobe para 59,71. Isso mudaria completamente a estimativa.

Vamos dar um exemplo do mundo real.

Em uma empresa de 50 funcionários, 45 pessoas com salário mensal de R $6.000,5 funcionários seniores com salário mensal de R$ 100.000 cada. Se você calcular o salário médio mensal dos funcionários da empresa é de R $15.400, o que lhe dará uma conclusão errada (a maioria dos funcionários tem salários, será R$
6.000, o que faz mais sentido do que a média. Por esta razão, a mediana é uma medida mais apropriada do que a média. Demonstrando o efeito do outlier.

Outlier é uma terminologia comumente usada por analistas e cientistas de dados, pois precisa de muita atenção, caso contrário pode resultar em estimativas totalmente erradas. Simplificando, Outlier é uma observação que parece distante e diverge de um padrão geral em uma amostra.

ORIGEM DOS OUTLIERS

* **Erros no input de dados**: - Erros humanos, como erros causados durante a coleta, gravação ou input de dados, podem causar valores discrepantes nos dados.

* **Erro de medição**: – É a fonte mais comum de outliers. Isso é causado quando o instrumento de medição utilizado apresenta defeito.

* **Outlier Natural**: - Quando um outlier não é artificial (devido a erro), é um outlier natural. A maioria dos dados do mundo real pertence a esta categoria.

IDENTIFICANDO OUTLIERS

Existem diversos métodos para identificar outliers em conjuntos de dados, cada um com suas vantagens e desvantagens. Sendo os mais comuns a Análise Visual, Testes Estatísticos e Algorítimos de detecção de outliers

DIFERENTE TÉCNICA DE IDENTIFICAÇÃO DE OUTLIERS.

1. Teste de Hipótese (Teste de Grubbs)

2. Método Z-score

3. Z-score Robusto

4. Método I.Q.R

5. Método de winsorização (Limite de Percentil)

6. Categorização DBSCAN

7. Isolation Forest

8. Visualização de dados

1. HYPOTHESIS TESTING (TESTE DE GRUBBS)

O teste de Grubbs, também conhecido como teste de Dixon unilateral, é um método estatístico utilizado para detectar outliers (valores atípicos) em conjuntos de dados unidimensionais. Ele se destaca por sua simplicidade, eficiência e robustez, tornando-se uma ferramenta valiosa em diversas áreas, como:

Análise de dados científicos: Identificar valores discrepantes em experimentos científicos.
Controle de qualidade: Detectar produtos defeituosos em manufaturas.
Análise financeira: Encontrar transações fraudulentas em cartões de crédito.
Análise de pesquisas: Eliminar valores extremos em pesquisas de opinião ou questionários.

Grubbs' test is defined for the hypothesis:
Ho: There are no outliers in the data set
H_1 : There is exactly one outlier in the data set

The Grubbs' test statistic is defined as:

$$G_{calculated} = \frac{\max|X_i - \overline{X}|}{SD}$$

with $\overline{X}$ and SD denoting the sample mean and standard deviation, respectively.

$$G_{critical} = \frac{(N-1)}{\sqrt{N}} \sqrt{\frac{\left(t_{\alpha/(2N),N-2}\right)^2}{N-2+\left(t_{\alpha/(2N),N-2}\right)^2}}$$

If the calculated value is greater than critical, you can reject the null hypothesis and concl

In [ ]:

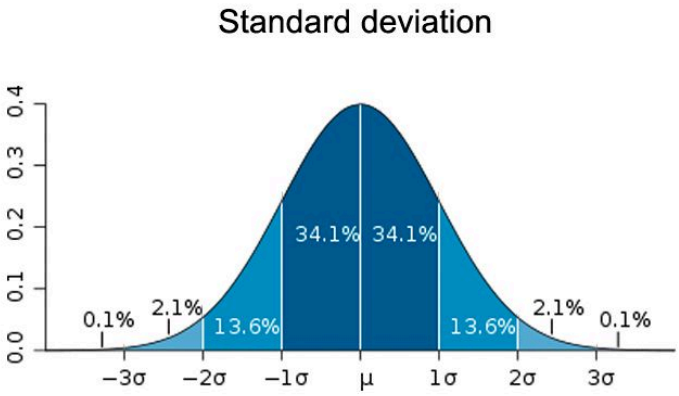
```
import numpy as np
import scipy.stats as stats
x = np.array([12,13,14,19,21,23])
y = np.array([12,13,14,19,21,23,45])
def grubbs_test(x):
    n = len(x)
    mean_x = np.mean(x)
    sd_x = np.std(x)
    numerator = max(abs(x-mean_x))
    g_calculated = numerator/sd_x
    print("Valor Calculado do Grubbs:",g_calculated)
    t_value = stats.t.ppf(1 - 0.05 / (2 * n), n - 2)
    g_critical = ((n - 1) * np.sqrt(np.square(t_value))) / (np.sqrt(n) * np.sqrt(n -
    print("Valor Crítico do Grubbs:",g_critical)
    if g_critical > g_calculated:
        print("De acordo com o teste de Grubbs, observamos que o valor calculado é m
```

```
    else:
        print("De acordo com o teste de Grubbs, observamos que o valor calculado é m
grubbs_test(x)
grubbs_test(y)
```

```
Valor Calculado do Grubbs: 1.4274928542926593
Valor Crítico do Grubbs: 1.8871451177787137
De acordo com o teste de Grubbs, observamos que o valor calculado é menors que o valo
r crítico, portanto aceitamos a hipótese nula e concluímos que não existe outlier

Valor Calculado do Grubbs: 2.2765147221587774
Valor Crítico do Grubbs: 2.019968507680656
De acordo com o teste de Grubbs, observamos que o valor calculado é maiior que o valo
r crítico, portanto rejeitamos a hipótese nula e concluímos que existe um outlier
```

2. MÉTODO Z-SCORE

Usando o método de Z-score, podemos identificar dados que estão X desvios padrões distantes da média.



% of values expected to lie in the symmetric interval (−zσ, zσ)

1σ	68.2689492%
2σ	95.4499736%
3σ	99.7300204%
4σ	99.993666%
5σ	99.9999426697%

A figura acima mostra a área sob a curva normal e a área que cada desvio padrão cobre.

* 68% dos pontos de dados estão entre + e - 1 desvio padrão.
* 95% dos pontos de dados estão entre + e - 2 desvios padrão
* 99.7% dos pontos de dados estão entre + e - 3 desvios padrão

---

Fórmula Z-score

$$Zscore = \frac{X - Mean}{StandardDeviation}$$

Se o Z-score de um dado for superior a 3 (99,7% da área), indica que o valor do dado é bastante diferente dos demais e portanto ser considerado outlier.

In [ ]:
```
import pandas as pd
import numpy as np

train = pd.read_csv('Data Analysis/HousePrices/dataset/train.csv')
out=[]
def Zscore_outlier(df):
    m = np.mean(df)
    sd = np.std(df)
    for i in df:
        z = (i-m)/sd
        if np.abs(z) > 3:
            out.append(i)
    print("Outliers:",out)
Zscore_outlier(train['LotArea'])
```

```
Outliers: [50271, 159000, 215245, 164660, 53107, 70761, 53227, 46589, 115149, 53504,
45600, 63887, 57200]
```

3. ROBUST Z-SCORE

Também é chamado de método de desvio absoluto da mediana. É semelhante ao método Z-score com algumas alterações nos parâmetros. Como a média e os desvios padrão são fortemente influenciados por outliers, alteramos esses parâmetros, usamos mediana e desvio absoluto da mediana.

Fórmula Robust Z-score

$$R.Z.score = \frac{0.6745*(X_i - Median)}{MAD}$$

Where MAD = median(|X-median|)

Suponha que x siga uma distribuição normal padrão. O MAD convergirá para a mediana da meia distribuição normal, que é o percentil 75% de uma distribuição normal, e N(0,75)≃0,6745.

In [ ]:
```
import pandas as pd
import numpy as np

train = pd.read_csv('Data Analysis/HousePrices/dataset/train.csv')
out=[]
def ZRscore_outlier(df):
    med = np.median(df)
    ma = stats.median_abs_deviation(df)
    for i in df:
        z = (0.6745*(i-med))/ (np.median(ma))
        if np.abs(z) > 3:
            out.append(i)
```

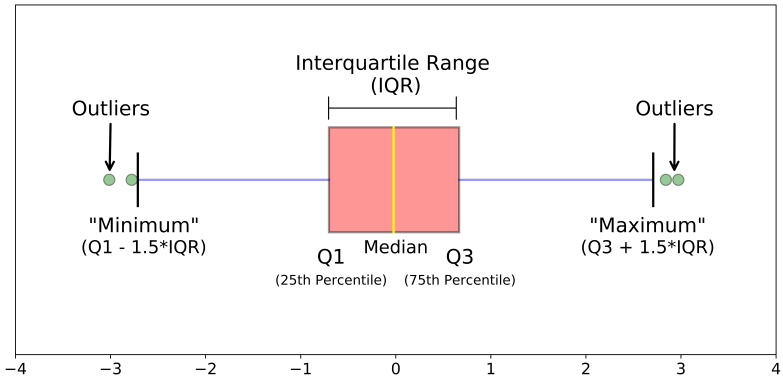
```
    print("Outliers:",out)
ZRscore_outlier(train['LotArea'])
```

```
Outliers: [50271, 19900, 21000, 21453, 19378, 31770, 22950, 25419, 159000, 19296, 391
04, 19138, 18386, 215245, 164660, 20431, 18800, 53107, 34650, 22420, 21750, 70761, 53
227, 40094, 32668, 21872, 21780, 25095, 46589, 20896, 18450, 21535, 26178, 115149, 21
695, 53504, 21384, 28698, 45600, 25286, 27650, 24090, 25000, 21286, 21750, 29959, 232
57, 35760, 35133, 32463, 18890, 24682, 23595, 36500, 63887, 20781, 25339, 57200, 2054
4, 19690, 21930, 26142]
```

4. MÉTODO IQR

Neste método detectamos outliers usando quartis. O IQR nos informa a variação no conjunto de dados. Qualquer valor que esteja além do intervalo de -1,5 x IQR a 1,5 x IQR são tratados como valores discrepantes

---



---

* Q1 representa o 1º quartil/25% dos dados.
* Q2 representa o 2º quartil/mediana/50% dos dados.
* Q3 representa o 3º quartil/75% dos dados.
* (Q1−1.5*IQR) representam o menor valor no conjunto de dados e (Q3+1.5*IQR) representam o maior valor no conjunto de dados.

In [ ]:

```
import pandas as pd
import numpy as np

train = pd.read_csv('Data Analysis/HousePrices/dataset/train.csv')
out=[]
def iqr_outliers(df):
    q1 = df.quantile(0.25)
    q3 = df.quantile(0.75)
    iqr = q3-q1
    Lower_tail = q1 - 1.5 * iqr
    Upper_tail = q3 + 1.5 * iqr
    for i in df:
        if i > Upper_tail or i < Lower_tail:
            out.append(i)
    print("Outliers:",out)
iqr_outliers(train['LotArea'])
```

```
Outliers: [50271, 19900, 21000, 21453, 19378, 31770, 22950, 25419, 159000, 19296, 391
04, 19138, 18386, 215245, 164660, 20431, 18800, 53107, 34650, 22420, 21750, 70761, 53
227, 40094, 32668, 21872, 21780, 25095, 46589, 20896, 18450, 21535, 26178, 115149, 21
```

```
695, 53504, 21384, 28698, 45600, 17920, 25286, 27650, 24090, 25000, 1300, 21286, 147
7, 21750, 29959, 18000, 23257, 17755, 35760, 18030, 35133, 32463, 18890, 24682, 2359
5, 17871, 36500, 63887, 20781, 25339, 57200, 20544, 19690, 21930, 26142]
```

5. WINSORIZATION METHOD(PERCENTILE CAPPING)

A windsorização, também conhecida como transformação de Tukey, é uma técnica estatística utilizada para lidar com outliers (valores atípicos) em conjuntos de dados. Ela consiste em substituir os outliers por valores mais próximos da mediana do conjunto, preservando a forma geral da distribuição dos dados e reduzindo o impacto desses valores extremos nas análises.

Este método é semelhante ao método IQR. Por exemplo, vamos identificar os outliers como os valores que estão entre 1% e 99% dos dados, os dados abaixo ou acima destes percentuais serão tratados como valores discrepantes.

Note que não fizemos a transformação dos dados que será tratado mais a frente.

In [ ]:
```
import pandas as pd
import numpy as np

train = pd.read_csv('Data Analysis/Titanic/dataset/train.csv')
out=[]
def Winsorization_outliers(df):
    q1 = np.percentile(df , 1)
    q3 = np.percentile(df , 99)
    for i in df:
        if i > q3 or i < q1:
            out.append(i)
    print("Outliers:",out)
Winsorization_outliers(train['Fare'])
```

```
Outliers: [263.0, 263.0, 512.3292, 262.375, 263.0, 263.0, 512.3292, 512.3292, 262.37
5]
```

6. DBSCAN (DENSITY-BASED SPATIAL CLUSTERING OF APPLICATIONS WITH NOISE)

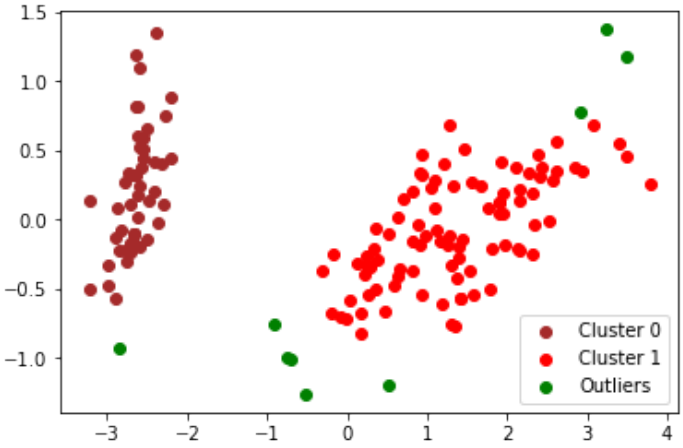
O DBSCAN (Density-Based Spatial Clustering of Applications with Noise) é um algoritmo robusto de agrupamento baseado em densidade, utilizado para identificar grupos de pontos em conjuntos de dados espaciais. Ele se destaca por sua capacidade de:
Encontrar clusters de diversas formas e tamanhos: Ao contrário de outros métodos que assumem formas predefinidas (como círculos ou elipses), o DBSCAN detecta clusters de formatos irregulares, adaptando-se aos dados.

Lidar com ruído e valores atípicos: O DBSCAN é capaz de identificar e separar pontos que não pertencem a nenhum cluster (ruído) do restante dos dados, tornando-o ideal para conjuntos de dados com outliers.
Funcionalidade intuitiva: O algoritmo opera com base em dois parâmetros simples: eps e MinPts, definindo a distância máxima entre pontos dentro de um mesmo cluster (eps) e o número mínimo de pontos necessários para formar um cluster (MinPts).

Cluster -1 indica que o cluster contém valores discrepantes e o restante dos clusters não possui valores discrepantes. Esta abordagem é semelhante ao agrupamento K-mean. Existem dois parâmetros necessários para DBSCAN. DBSCAN fornece o melhor resultado para detecção de valores discrepantes multivariados.

---

1. epsilon: um parâmetro de distância que define o raio onde procurar vizinhos próximos.
2. quantidade mínima de pontos necessários para formar um cluster.



Funcionamento do DBSCAN:

Pontos Centrais e Pontos de Borda: O DBSCAN inicia identificando os pontos centrais: pontos que possuem pelo menos MinPts pontos dentro de uma vizinhança circular de raio eps. Já os pontos de borda são aqueles que estão dentro da vizinhança de um ponto central, mas não possuem MinPts pontos em sua própria vizinhança.
Expansão de Clusters: Cada ponto central forma um cluster inicial. Em seguida, o algoritmo expande esses clusters iterativamente, agregando pontos de borda que estão dentro da vizinhança de pontos já pertencentes ao cluster.
Ruído: Pontos que não podem ser agregados a nenhum cluster durante a expansão são classificados como ruído.

In [ ]:
```
from sklearn.cluster import DBSCAN

train = pd.read_csv('Data Analysis/Titanic/dataset/train.csv')
def DB_outliers(df):
    outlier_detection = DBSCAN(eps = 2, metric='euclidean', min_samples = 5)
    clusters = outlier_detection.fit_predict(df.values.reshape(-1,1))
    data = pd.DataFrame()
    data['cluster'] = clusters
    #print(data['cluster'].value_counts().sort_values(ascending=False))
    print(data['cluster'].value_counts())
DB_outliers(train['Fare'])
```

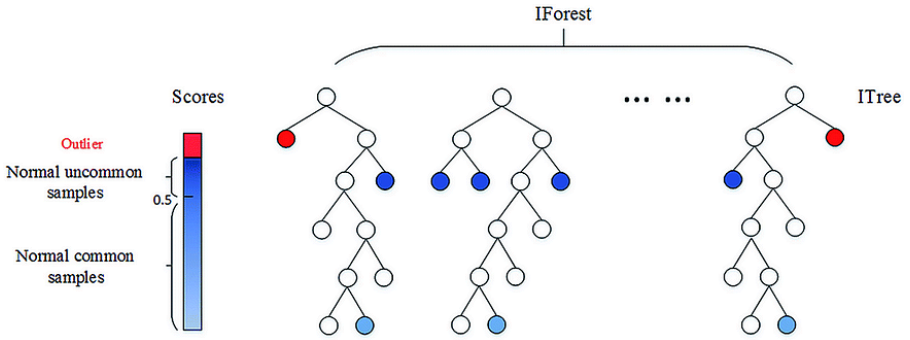
```
 0     705
 2      50
 4      36
-1      32
 6      15
 1      12
 7       8
 5       7
 8       7
 9       7
 3       6
 10      6
Name: cluster, dtype: int64
```

In [ ]:
```
train.shape
```

Out[ ]:
```
(891, 12)
```

7. ISOLATION FOREST OU FLORESTA DE ISOLAMENTO

É é um algoritmo de aprendizado de máquina não supervisionado utilizado para detectar anomalias em conjuntos de dados. Ele se destaca por sua simplicidade, eficiência e robustez, tornando-o uma ferramenta valiosa em diversas áreas, como:
Detecção de Fraude: Identificar transações fraudulentas em cartões de crédito ou seguros.
Detecção de Intruções: Detectar atividades maliciosas em redes de computadores.
Manutenção Preditiva: Prever falhas em equipamentos industriais.
Análise de Imagens: Identificar objetos defeituosos em produtos manufaturados.
Detecção de Anomalias Médicas: Encontrar padrões anormais em exames médicos.



Como Funciona o Isolation Forest:

Construção de Árvores Aleatórias: O algoritmo constrói um conjunto de árvores de decisão de forma aleatória, dividindo os dados em subconjuntos a cada nó. Em cada divisão, um recurso e um valor de divisão são selecionados aleatoriamente.
Isolamento de Anomalias: O Isolation Forest isola as anomalias mais rapidamente do que as observações normais. Em cada nível da árvore, o caminho percorrido para isolar um ponto é registrado. Pontos que são isolados em níveis mais rasos da árvore são considerados mais propensos a serem anomalias.
Pontuação de Anomalia: Cada ponto recebe uma pontuação de anomalia com base no comprimento médio do caminho percorrido para isolá-lo em todas as árvores. Pontos com pontuações mais altas são considerados mais propensos a serem anomalias.

No modelo utilizado abaixo, se o resultado for -1, significa que este ponto de dados específico é um valor discrepante. Se o resultado for 1, significa que o ponto de dados não é um valor discrepante.

In [ ]:
```
from sklearn.ensemble import IsolationForest
import numpy as np
import pandas as pd

train = pd.read_csv('Data Analysis/Titanic/dataset/train.csv')
train['Fare'].fillna(train[train.Pclass==3]['Fare'].median(),inplace=True)
def Iso_outliers(df):
    iso = IsolationForest(random_state = 1, contamination= 'auto')
    preds = iso.fit_predict(df.values.reshape(-1,1))
    data = pd.DataFrame()
    data['cluster'] = preds
    print(data['cluster'].value_counts().sort_values(ascending=False))
Iso_outliers(train['Fare'])
```

```
 1    706
-1    185
Name: cluster, dtype: int64
```

8. VISUALIZAÇÃO DE DADOS

A visualização de dados é útil para limpeza de dados, exploração de dados, detecção de valores discrepantes e grupos incomuns, identificação de tendências e clusters, etc. Aqui está a lista de gráficos de visualização de dados para detectar valores discrepantes.

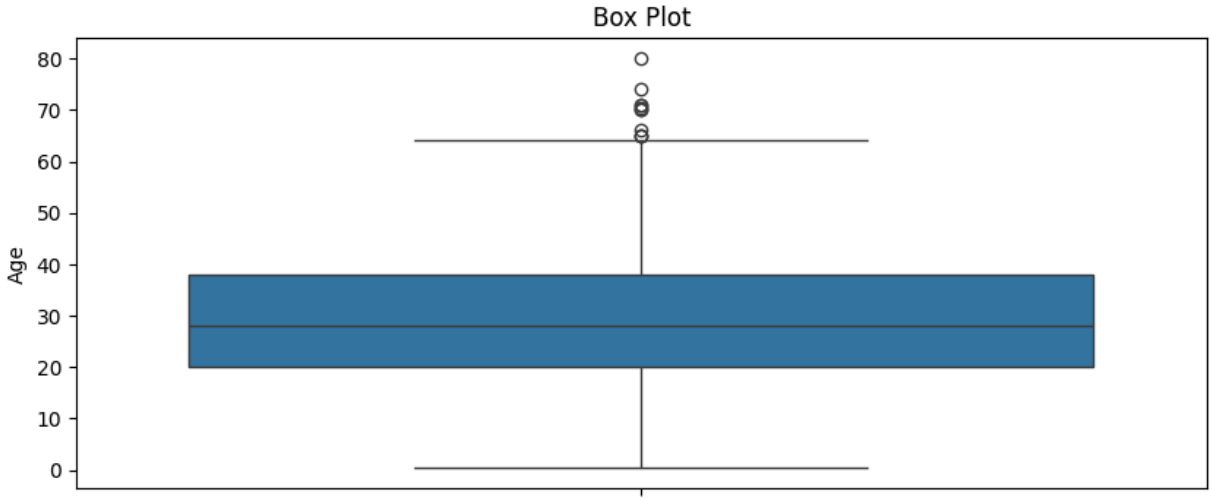
---

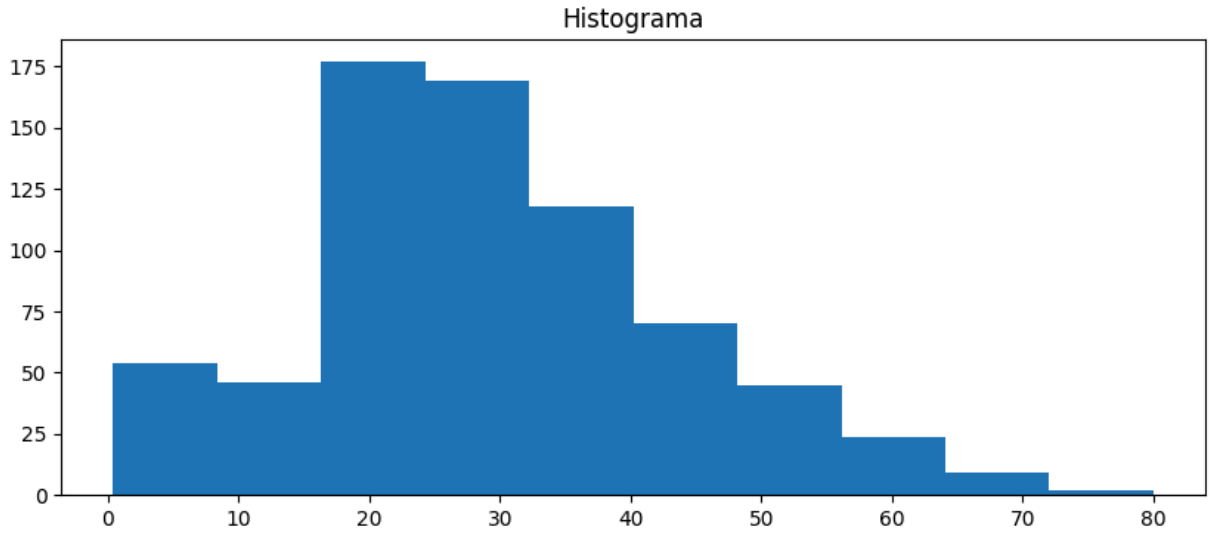
1. Box and whisker plot (box plot).
2. Gráfico de dispersão.

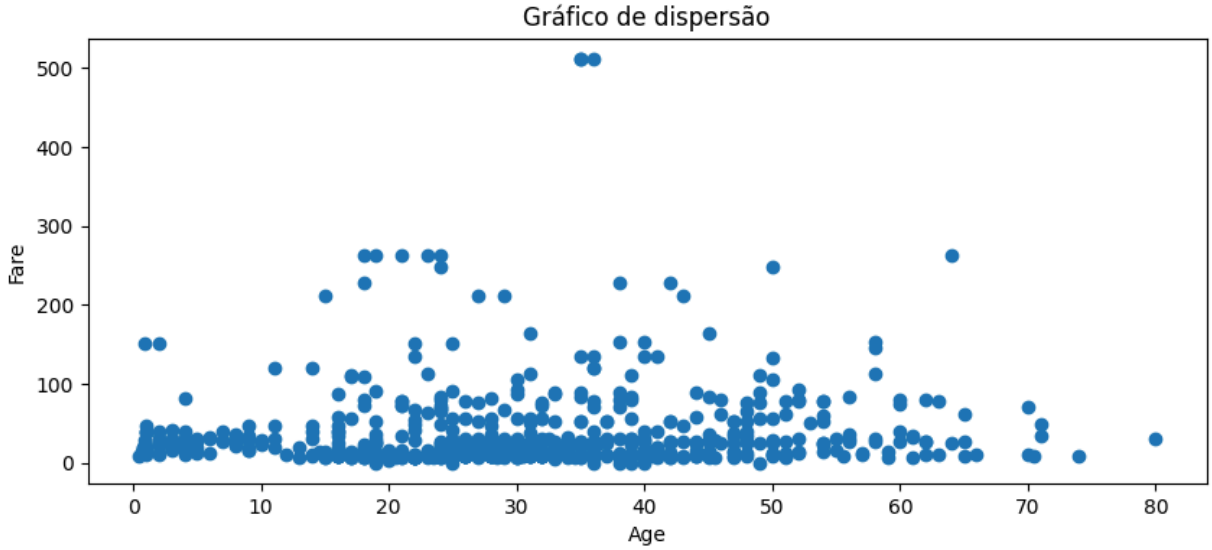
3. Histograma.
4. Gráfico de distribuição.
5. QQ plot.

In [ ]:

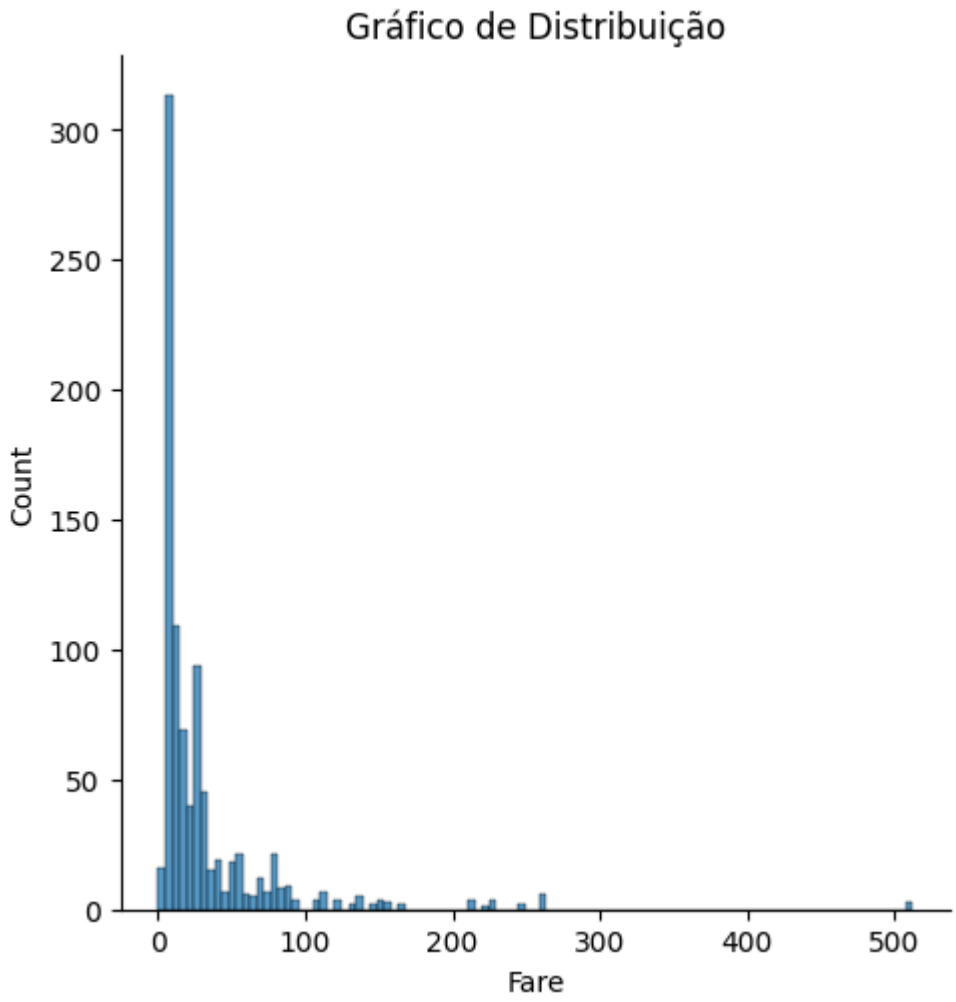
```
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
from statsmodels.graphics.gofplots import qqplot

train = pd.read_csv('Data Analysis/Titanic/dataset/train.csv')
def Box_plots(df):
    plt.figure(figsize=(10, 4))
    plt.title("Box Plot")
    sns.boxplot(df)
    plt.show()
Box_plots(train['Age'])

def hist_plots(df):
    plt.figure(figsize=(10, 4))
    plt.hist(df)
    plt.title("Histograma")
    plt.show()
hist_plots(train['Age'])

def scatter_plots(df1,df2):
    fig, ax = plt.subplots(figsize=(10,4))
    ax.scatter(df1,df2)
    ax.set_xlabel('Age')
    ax.set_ylabel('Fare')
    plt.title("Gráfico de dispersão")
    plt.show()
scatter_plots(train['Age'],train['Fare'])

def dist_plots(df):
    plt.figure(figsize=(10, 4))
    sns.displot(df)
    plt.title("Gráfico de Distribuição")
    sns.despine()
    plt.show()
dist_plots(train['Fare'])

def qq_plots(df):
    plt.figure(figsize=(10, 4))
    qqplot(df,line='s')
    plt.title("Normal QQPlot")
    plt.show()
qq_plots(train['Fare'])
```



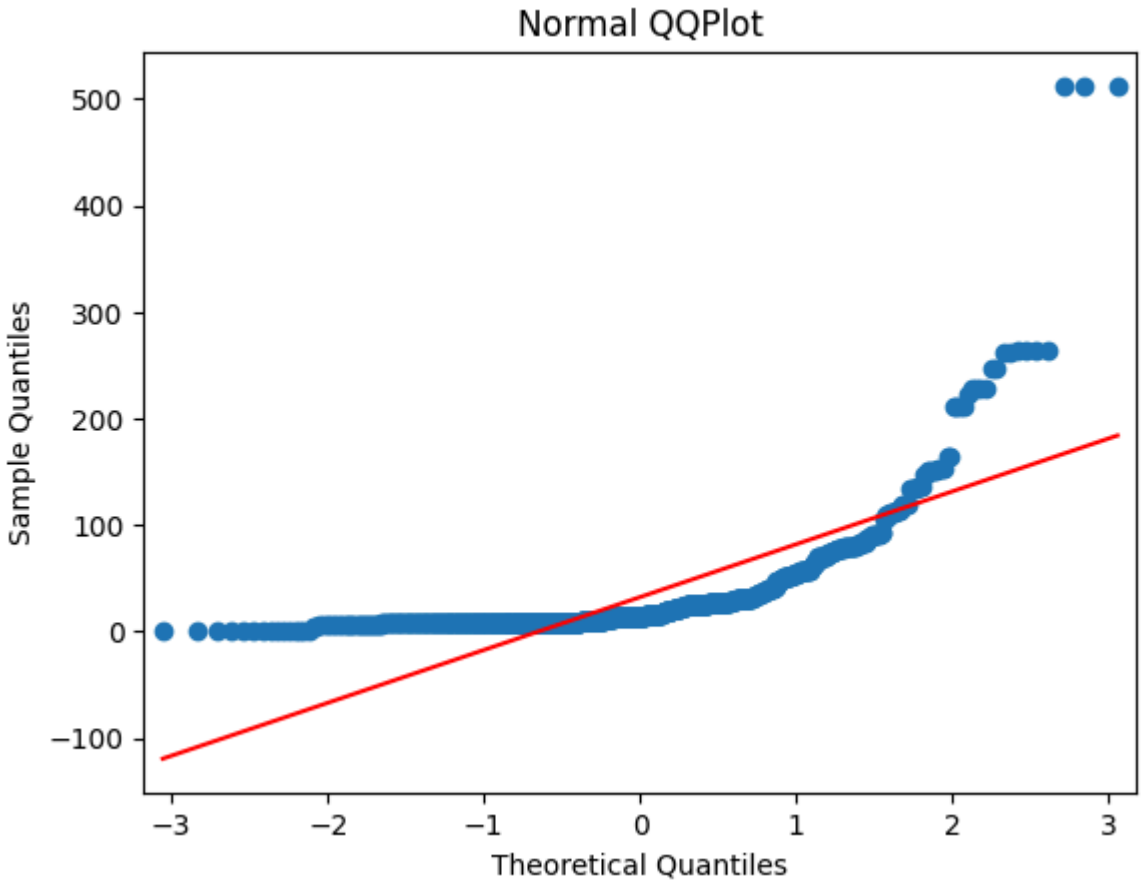





```
<Figure size 1000x400 with 0 Axes>
```



<Figure size 1000x400 with 0 Axes>



TRATAMENTO

Depois de detectar os outliers, devemos removê-los ou tratá-los

---

A presença de outliers em um conjunto de dados pode ter um impacto significativo nas análises e conclusões estatísticas. Eles podem:

- **Distorcer a média e outras medidas de tendência central**: Outliers podem fazer com que a média, mediana e outras medidas de tendência central representem mal o "valor típico" dos dados.

- **Aumentar a variância e outras medidas de dispersão**: Outliers podem aumentar a variância, desvio padrão e outras medidas de dispersão, fazendo com que os dados pareçam mais dispersos do que realmente são

- **Afetar a significância estatística dos testes**: Outliers podem influenciar indevidamente os resultados de testes estatísticos, levando a conclusões errôneas.

A maioria dos algoritmos de aprendizado de máquina não funciona bem na presença de valores discrepantes. Portanto, é desejável detectar e remover valores discrepantes.

Por todas essas razões, devemos ter cuidado com os valores discrepantes e tratá-los antes de construir um modelo estatístico/de aprendizado de máquina. Existem algumas técnicas usadas para lidar com outliers.

---

1. Exclusão.

2. Transformação.

3. Substituição.

4. Tratamento Separado

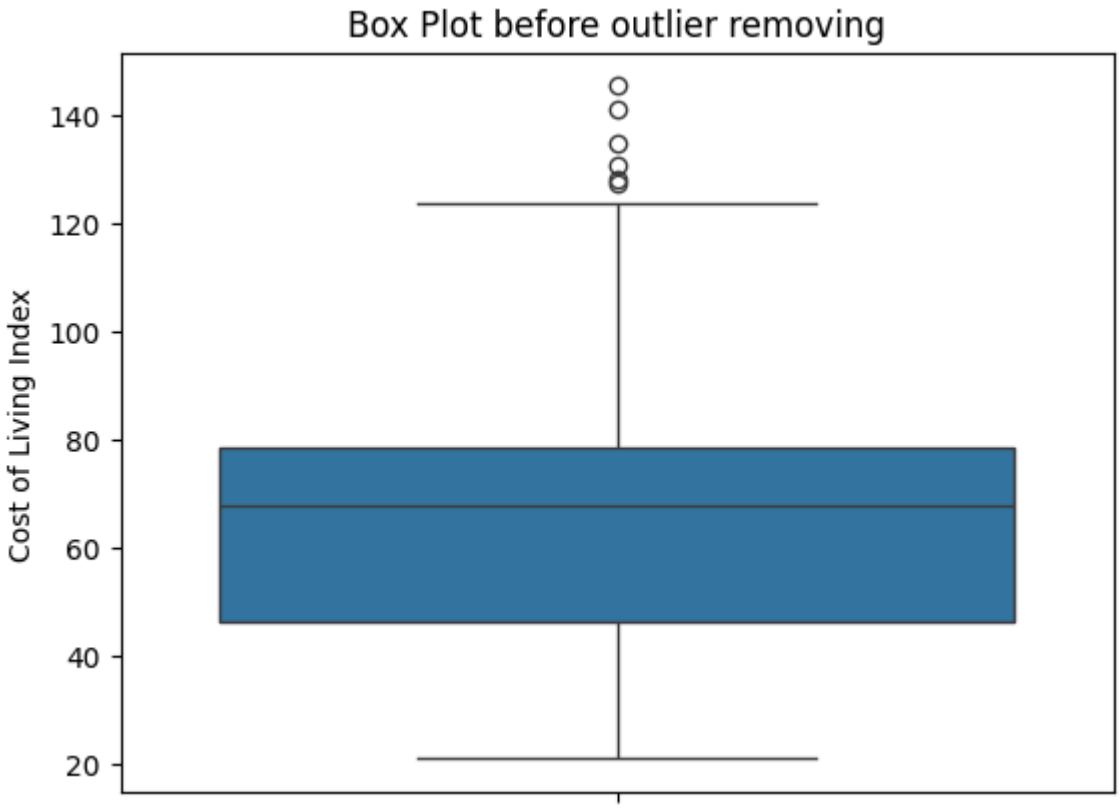
EXCLUSÃO:

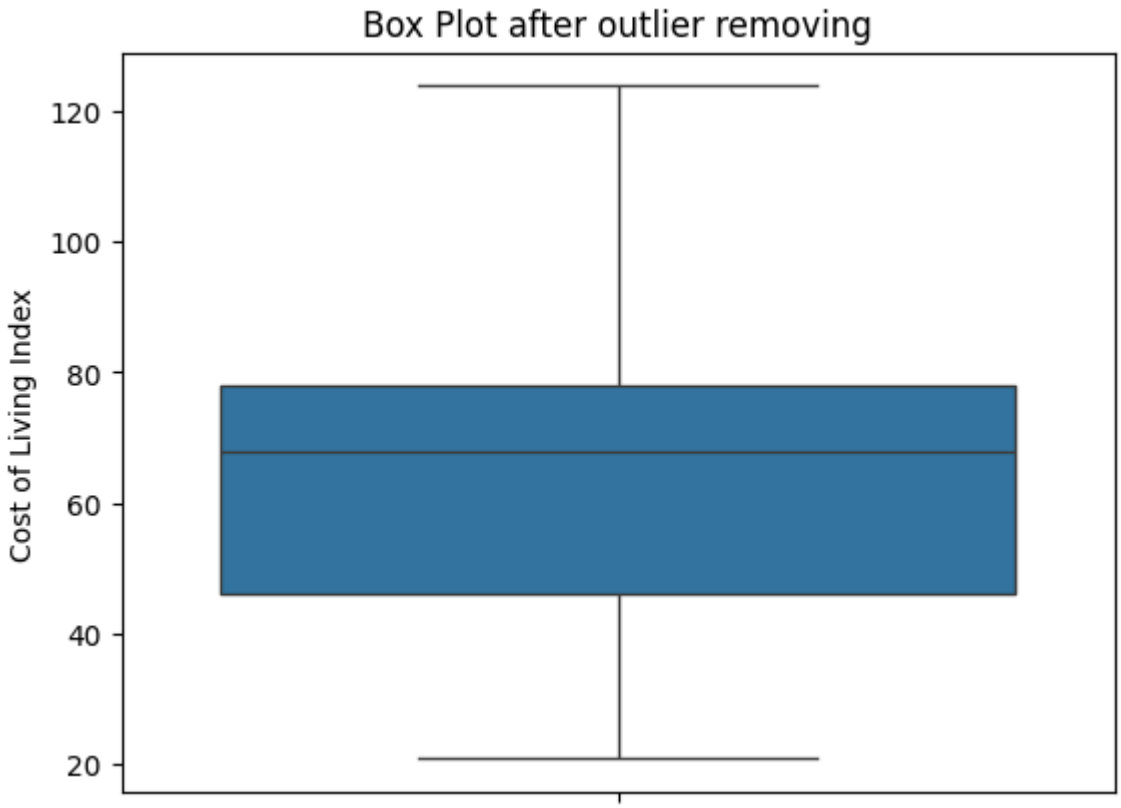
Excluímos valores discrepantes se for devido a erro de entrada de dados, erro de processamento de dados ou observações discrepantes muito pequenas em quantidade. Também podemos usar o corte em ambas as extremidades para remover valores discrepantes. Mas excluir a observação não é uma boa ideia quando temos um conjunto de dados pequeno.

É importante remover apenas os outliers genuínos e não excluir dados válidos.

In [ ]:

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

train = pd.read_csv('Data Analysis/CostofLiving/dataset/cost-of-living-2018.csv')
sns.boxplot(train['Cost of Living Index'])
plt.title("Box Plot before outlier removing")
plt.show()
def drop_outliers(df, field_name):
    iqr = 1.5 * (np.percentile(df[field_name], 75) - np.percentile(df[field_name], 2
    df.drop(df[df[field_name] > (iqr + np.percentile(df[field_name], 75))].index, in
    df.drop(df[df[field_name] < (np.percentile(df[field_name], 25) - iqr)].index, in
drop_outliers(train, 'Cost of Living Index')
sns.boxplot(train['Cost of Living Index'])
plt.title("Box Plot after outlier removing")
plt.show()
```





TRANSFORMAÇÃO:

A transformação de variáveis também pode eliminar valores discrepantes. Esses valores transformados reduzem a variação causada por valores extremos.

---

1. Dimensionamento
2. Transformação Logarítimica
3. Raiz Cúbica
4. Box-Cox transformation

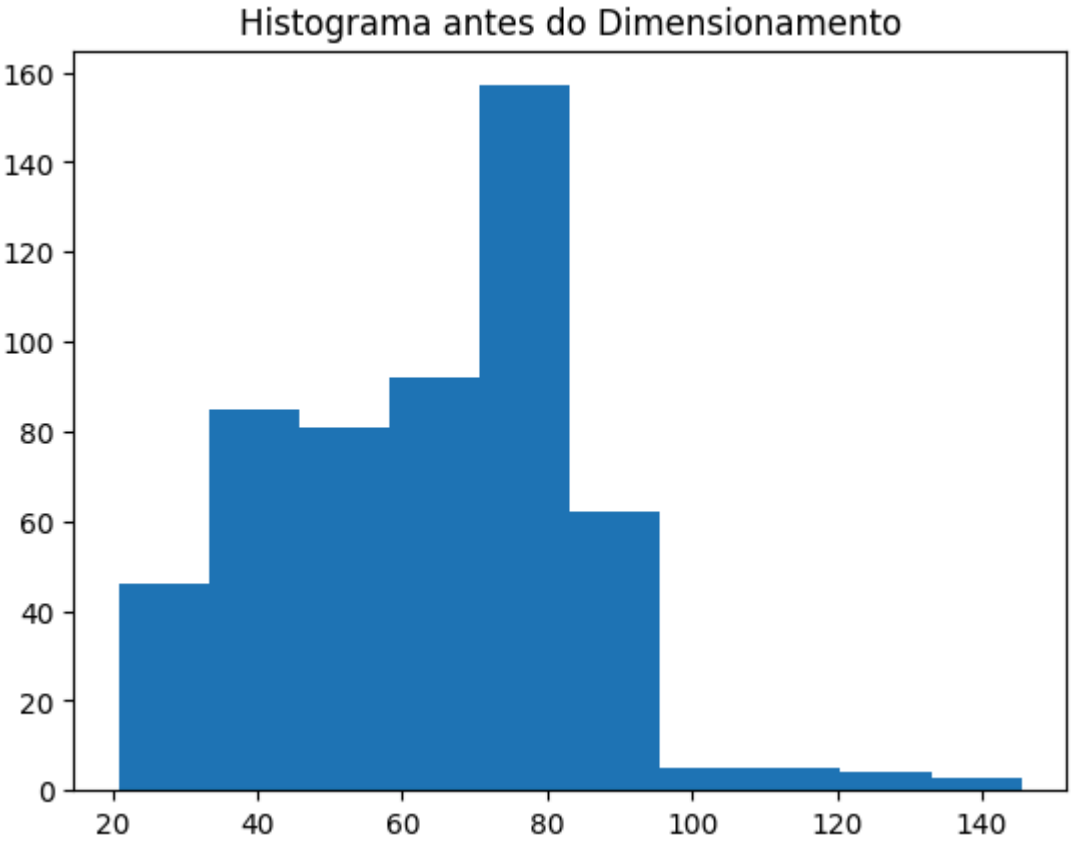
---

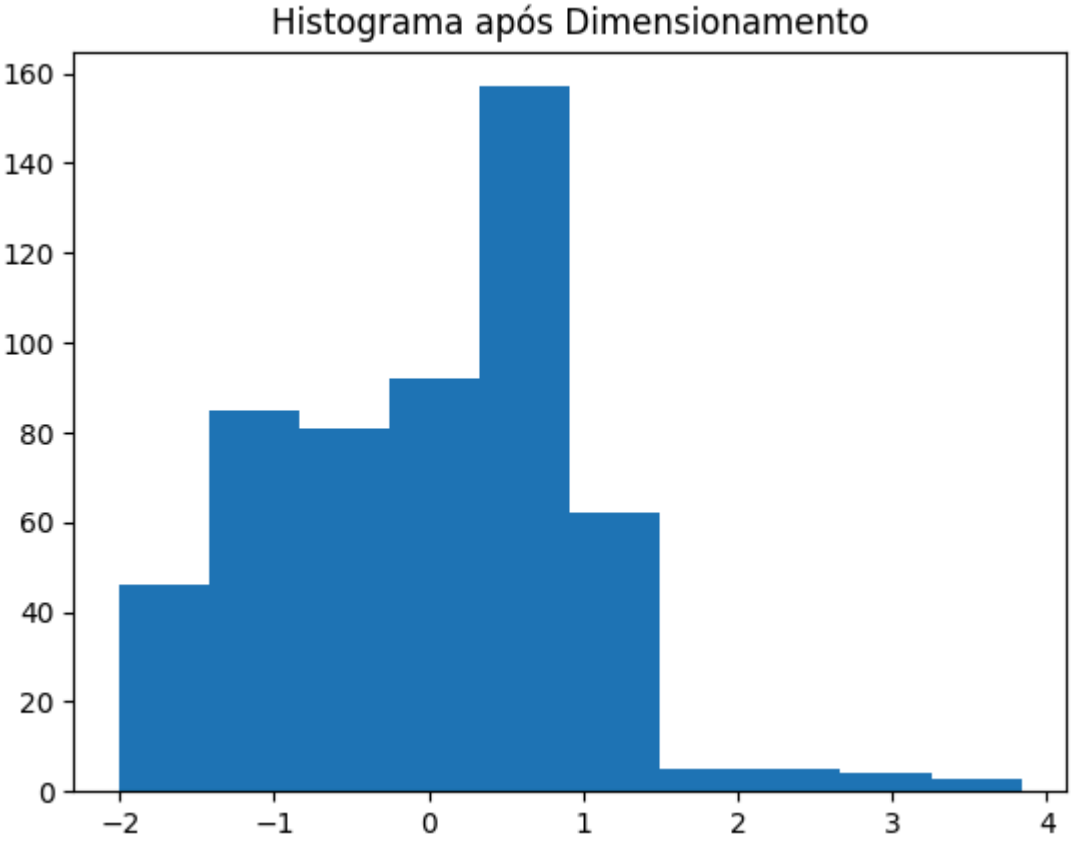
* Essas técnicas convertem valores no conjunto de dados em valores menores.
* Se os dados tiverem muitos valores extremos ou distorcidos, este método ajuda a normalizar seus dados.
* Mas essas técnicas nem sempre oferecem os melhores resultados.
* Não há perda de dados com esses métodos.
* Dentre todos esses métodos, a transformação boxcox dá o melhor resultado.

In [ ]:

```
#Scalling - Dimensionamento
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
from sklearn import preprocessing

train = pd.read_csv('Data Analysis/CostofLiving/dataset/cost-of-living-2018.csv')
```

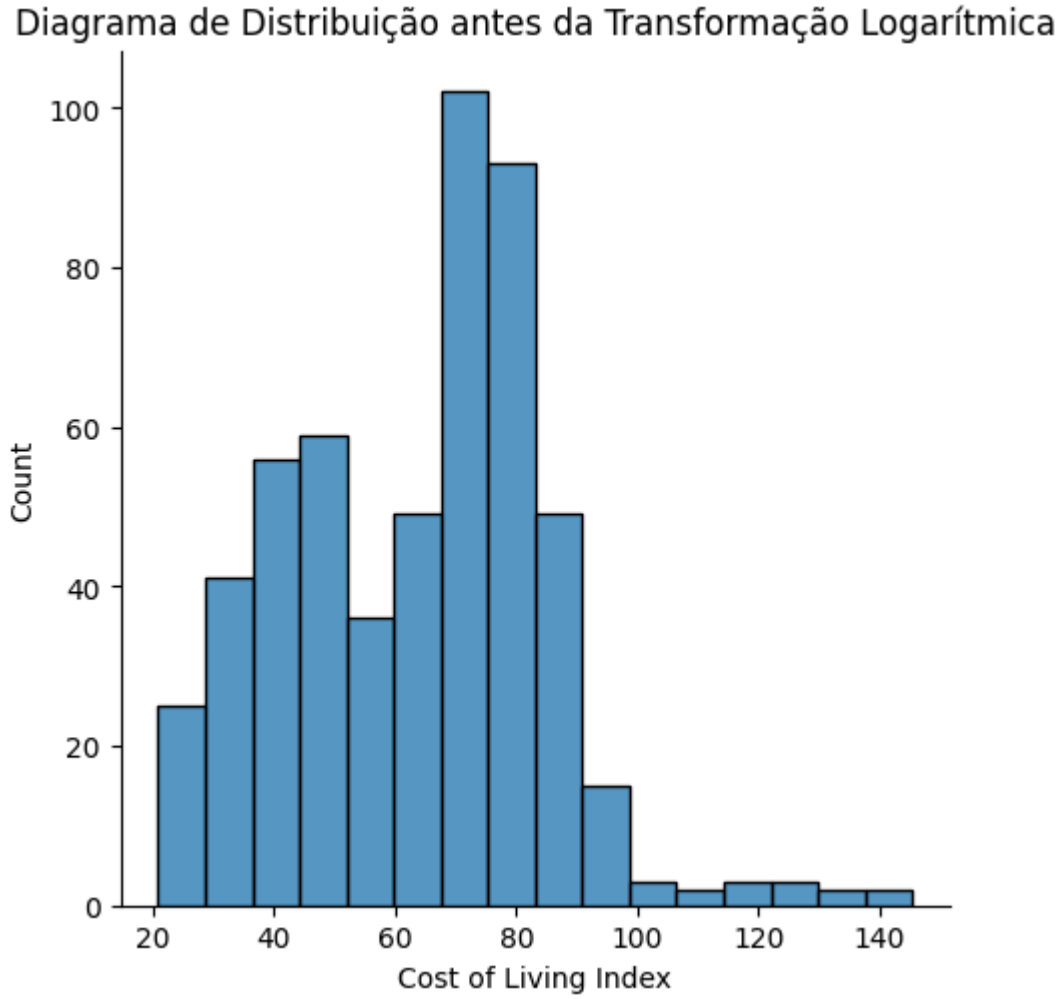
```
plt.hist(train['Cost of Living Index'])
plt.title("Histograma antes do Dimensionamento")
plt.show()
scaler = preprocessing.StandardScaler()
train['Cost of Living Index'] = scaler.fit_transform(train['Cost of Living Index'].v
plt.hist(train['Cost of Living Index'])
plt.title("Histograma após Dimensionamento")
plt.show()
```

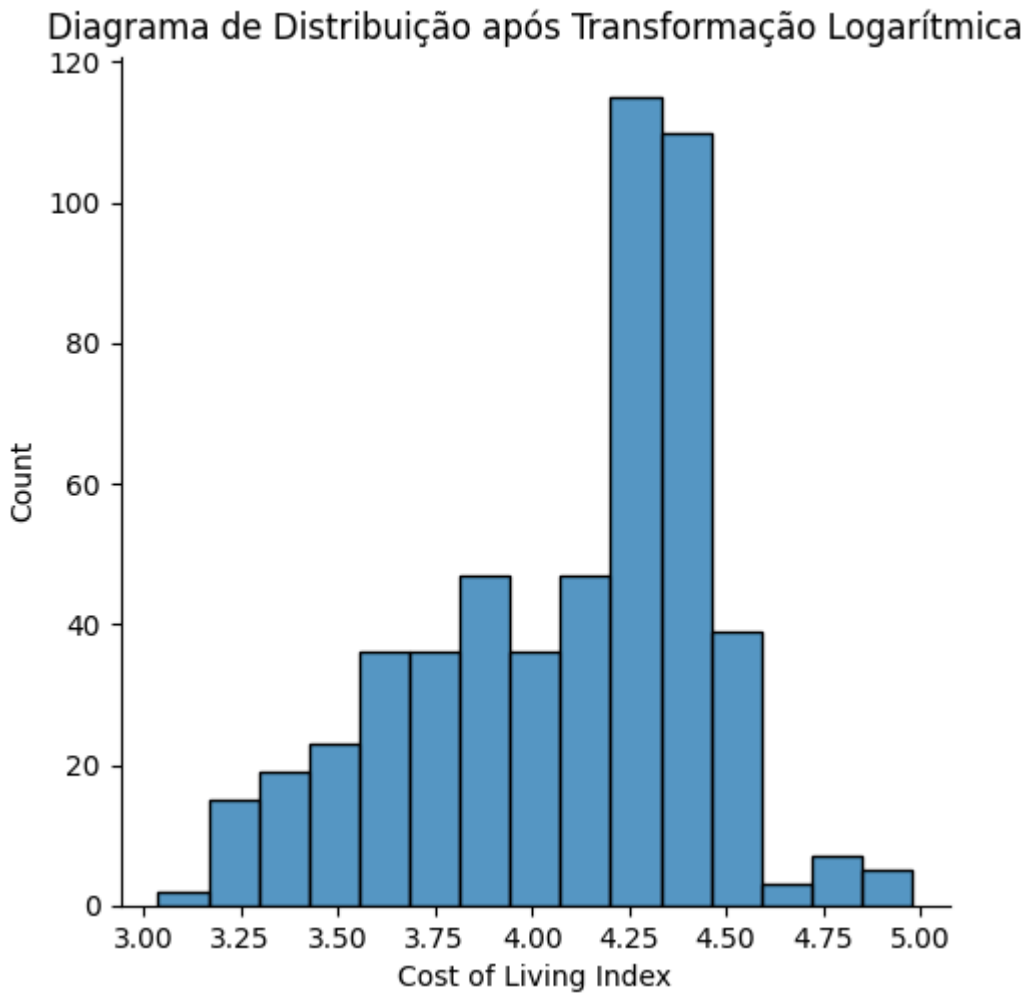




In [ ]:

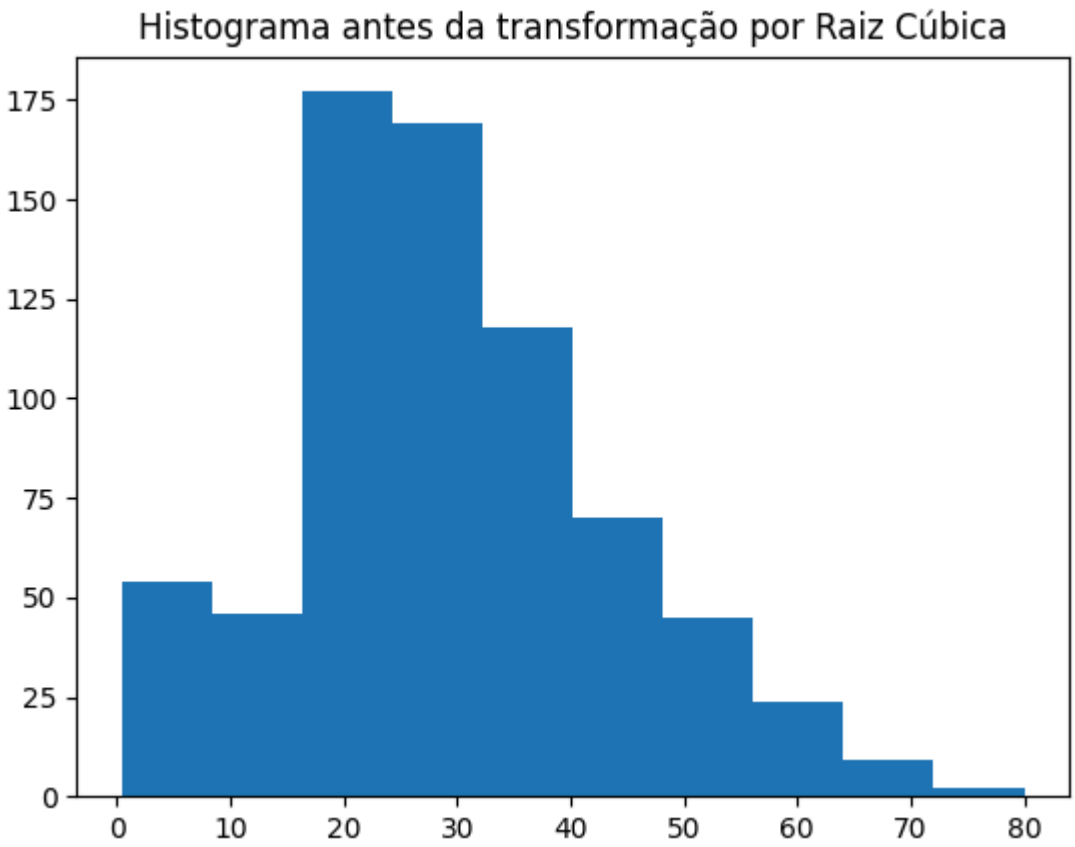
```
#Log Transformation - Transformação Logarítimica
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt

train = pd.read_csv('Data Analysis/CostofLiving/dataset/cost-of-living-2018.csv')
sns.displot(train['Cost of Living Index'])
plt.title("Diagrama de Distribuição antes da Transformação Logarítmica")
sns.despine()
plt.show()
train['Cost of Living Index'] = np.log(train['Cost of Living Index'])
sns.displot(train['Cost of Living Index'])
plt.title("Diagrama de Distribuição após Transformação Logarítmica")
sns.despine()
plt.show()
```

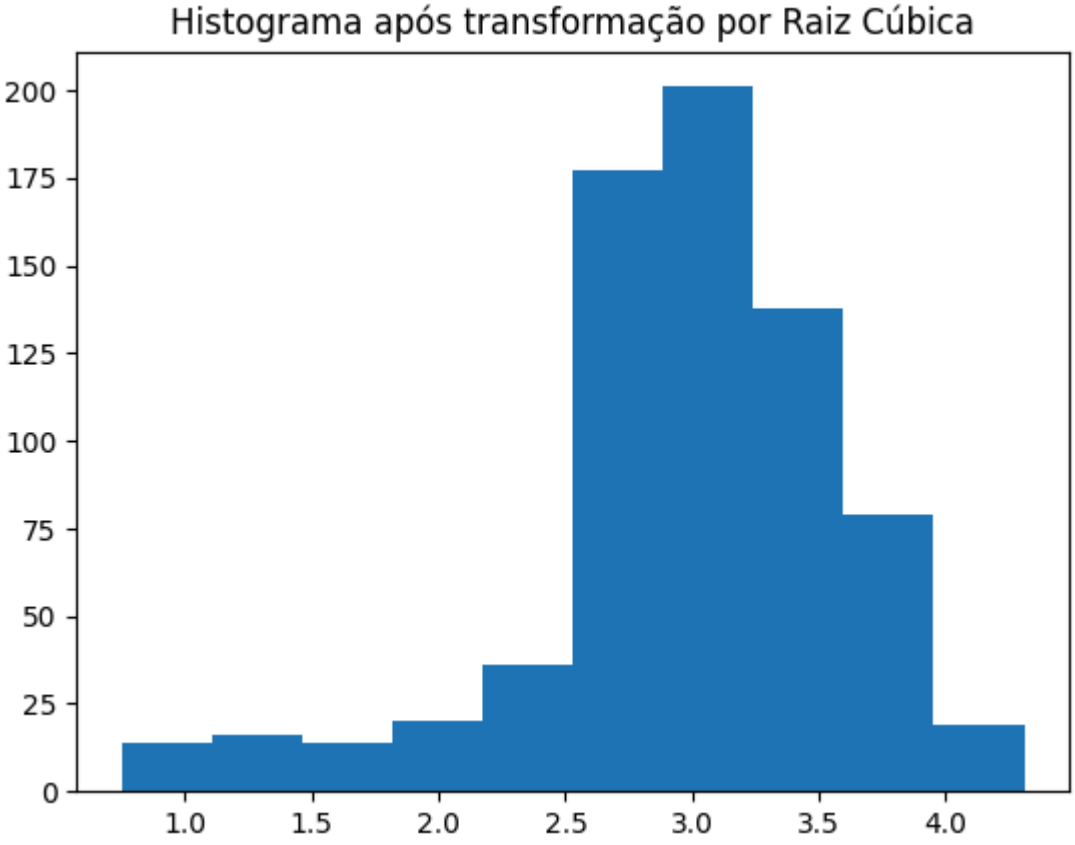




In [ ]:

```
#cube root Transformation - Raiz Cúbica
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt

train = pd.read_csv('Data Analysis/Titanic/dataset/train.csv')
plt.hist(train['Age'])
plt.title("Histograma antes da transformação por Raiz Cúbica")
plt.show()
train['Age'] = (train['Age']**(1/3))
plt.hist(train['Age'])
plt.title("Histograma após transformação por Raiz Cúbica")
plt.show()
```

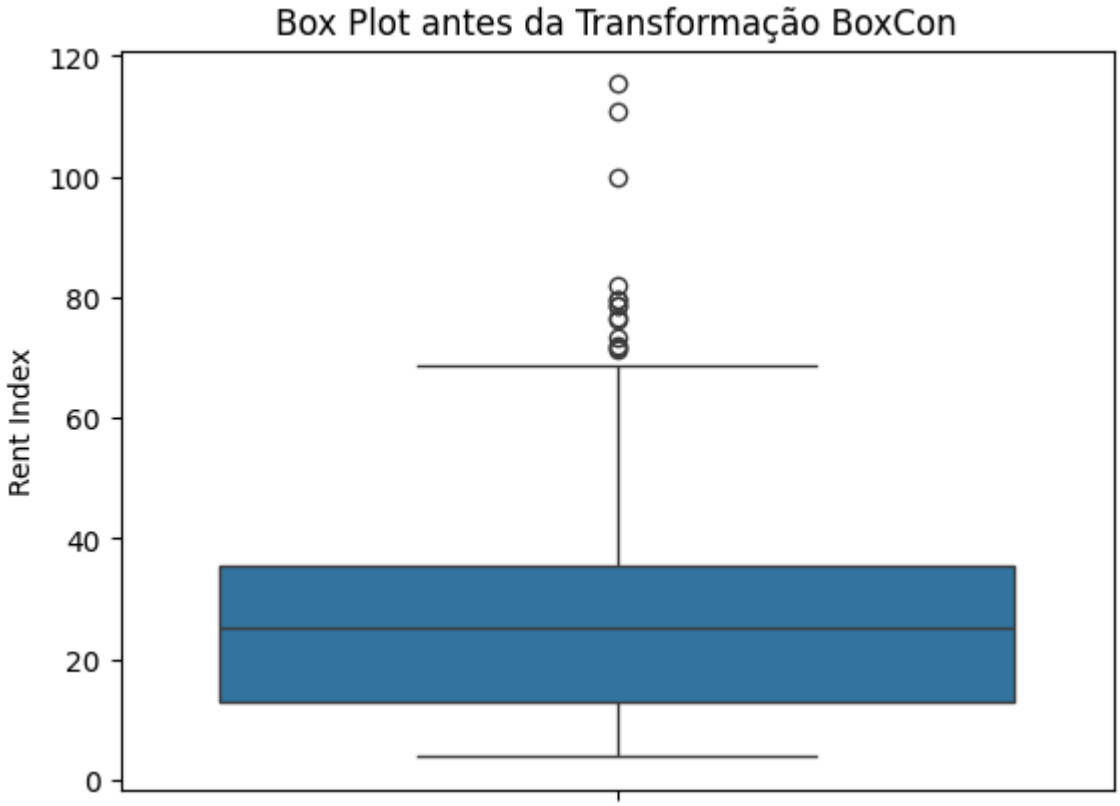


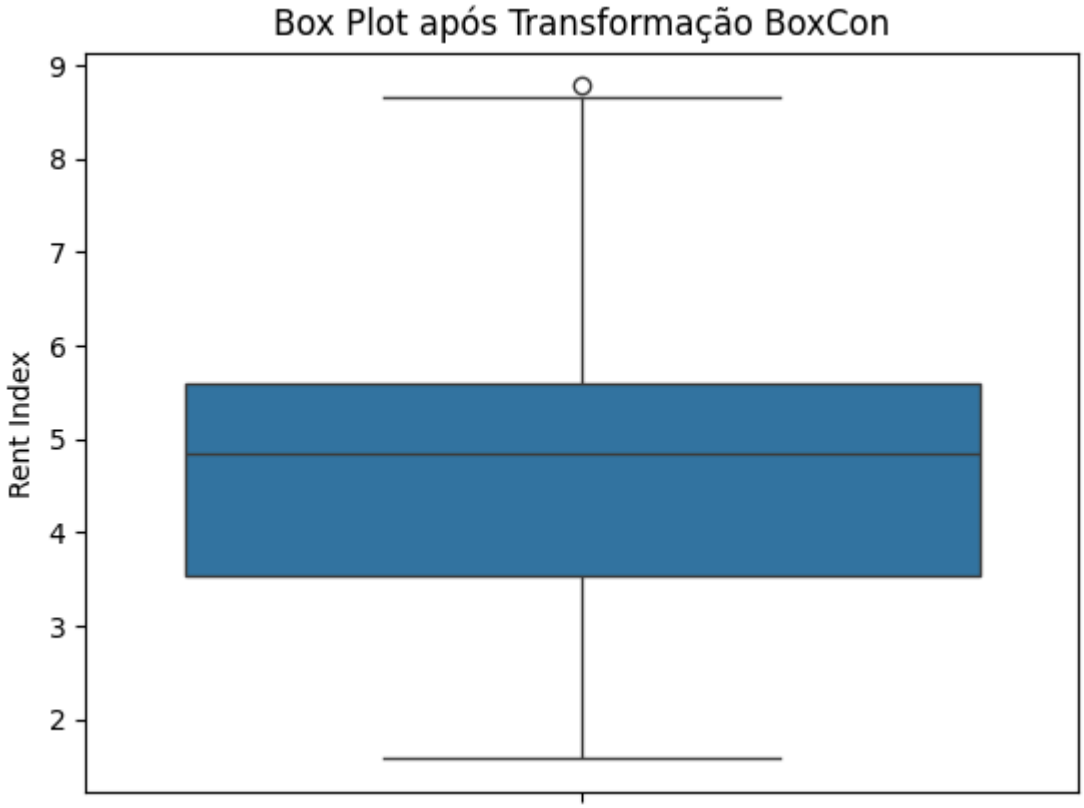


In [ ]:

```
#Box-transformation
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
import scipy

train = pd.read_csv('Data Analysis/CostofLiving/dataset/cost-of-living-2018.csv')
```

```
sns.boxplot(train['Rent Index'])
plt.title("Box Plot antes da Transformação BoxCon")
plt.show()
train['Rent Index'],fitted_lambda= scipy.stats.boxcox(train['Rent Index'] ,lmbda=Non
sns.boxplot(train['Rent Index'])
plt.title("Box Plot após Transformação BoxCon")
plt.show()
```





SUBSTITUIÇÃO

Podemos substituir de valores ausentes bem como substituir valores discrepantes usando média, mediana ou valor zero. Como estamos substituindo, não há perda de dados. Aqui a mediana é apropriada porque não é afetada por valores discrepantes.

Substituir outliers em conjuntos de dados por valores como mediana ou média antes de treinar modelos de machine learning é uma prática comum, mas que exige cautela e compreensão de seus impactos. Essa técnica, também conhecida como imputação, pode influenciar significativamente o desempenho dos modelos e a interpretação dos resultados.

Analisando os Impactos:

Redução do Efeito dos Outliers: Substituir outliers por valores mais próximos da distribuição central minimiza o impacto desses valores extremos nos cálculos das estatísticas do conjunto de dados. Isso pode ser crucial para evitar distorções na média, mediana, desvio padrão e outros indicadores, que por sua vez influenciam o treinamento dos modelos.
Melhoria da Previsão: Ao reduzir o efeito dos outliers, a precisão das previsões dos modelos pode ser melhorada, especialmente em tarefas sensíveis a valores extremos. Isso porque o modelo se concentra nos padrões da maioria dos dados, ignorando os outliers que podem levar a previsões incorretas.
Estabilidade do Treinamento: Substituir outliers pode estabilizar o processo de treinamento, evitando que o modelo se ajuste excessivamente a esses valores atípicos e melhore a generalização do modelo para novos dados.
Mascaramento de Informação: No entanto, essa técnica também pode mascarar informações valiosas contidas nos outliers. Se esses valores forem relevantes para a análise, removê-los ou substituí-los pode levar a conclusões errôneas ou à perda de insights importantes.
Dificuldade na Interpretação: A interpretação dos resultados dos modelos pode ser dificultada pela substituição de outliers, pois os valores originais não estão presentes nos dados utilizados para o treinamento. Isso torna crucial a documentação da estratégia de imputação e a consideração dos valores originais na análise final.

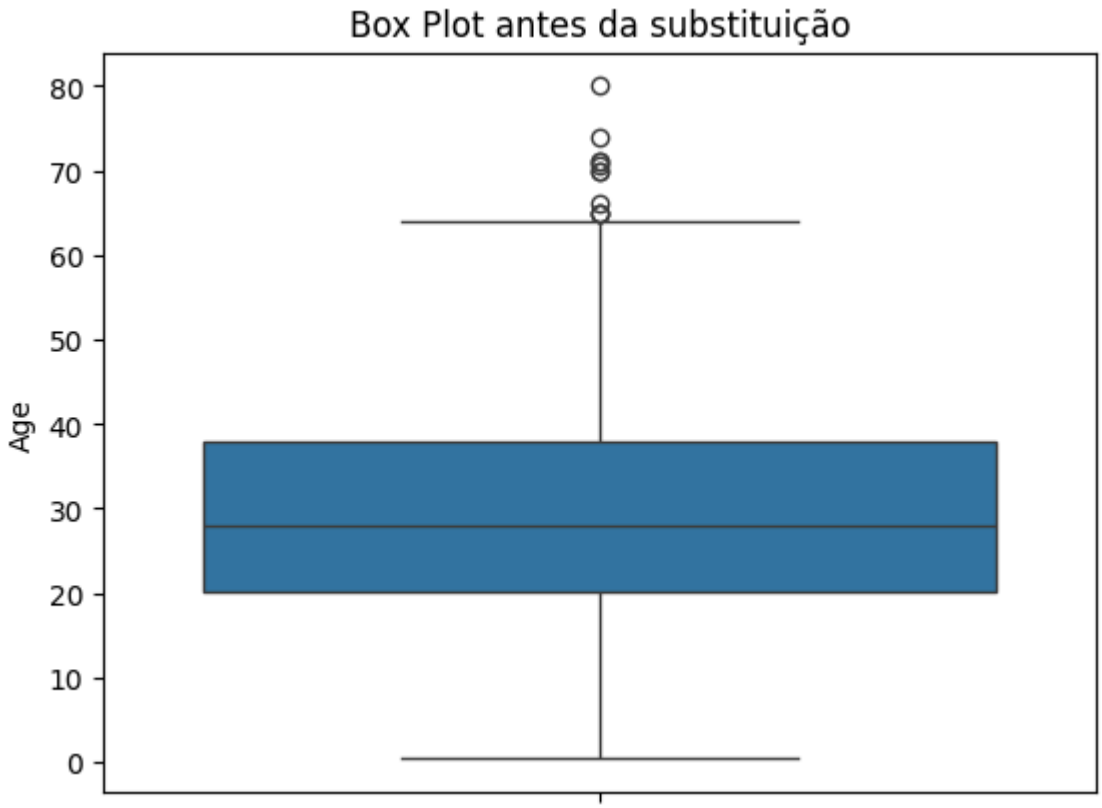
ATENÇÃO: Lidar com valores ausentes em conjuntos de dados é um desafio comum no machine learning. Substituir esses valores por técnicas como imputação pode ser crucial para o sucesso do modelo, mas é importante entender os impactos dessa prática e escolher a estratégia adequada.

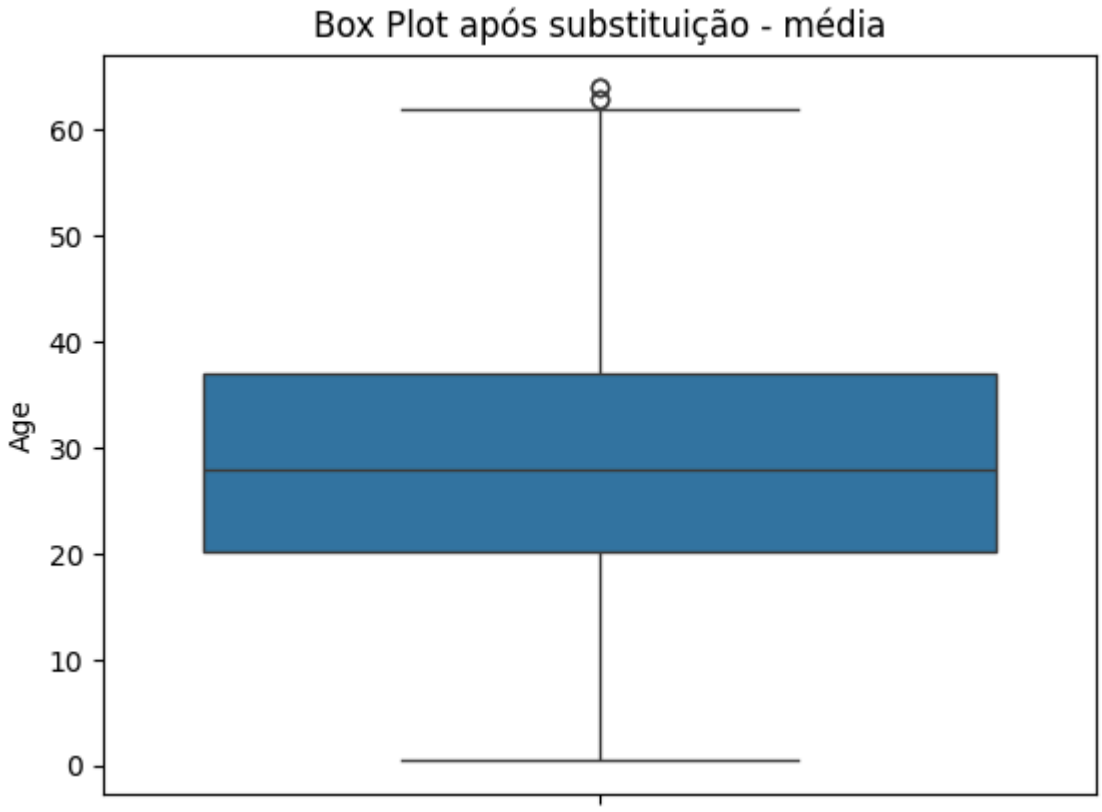
Impactos da Substituição de Valores Ausentes:

Redução do Viés: Substituir valores ausentes reduz o viés que esses valores podem introduzir nos cálculos de estatísticas do conjunto de dados, como média, mediana e desvio padrão. Isso pode ser fundamental para evitar distorções que afetam o treinamento dos modelos.
Melhoria do Desempenho: Ao reduzir o viés, a precisão e a generalização dos modelos podem ser melhoradas, especialmente em tarefas sensíveis à falta de dados. O modelo se concentra nos padrões dos dados presentes, ignorando os valores ausentes que podem levar a previsões incorretas.
Maior Estabilidade do Treinamento: Substituir valores ausentes pode estabilizar o processo de treinamento, evitando que o modelo se ajuste excessivamente à falta de informação em alguns pontos e melhorando a convergência do algoritmo.
Possibilidade de Introdução de Novos Erros: No entanto, a imputação também pode introduzir novos erros nos dados, se não for realizada de forma adequada. A escolha da técnica de imputação e dos valores utilizados é crucial para minimizar o impacto negativo na qualidade dos dados.
Dificuldade na Interpretação: A interpretação dos resultados dos modelos pode ser dificultada pela imputação, pois os valores originais não estão presentes nos dados utilizados para o treinamento. Isso torna crucial a documentação da estratégia de imputação e a consideração dos valores originais na análise final.

In [ ]:

```
#mean imputation
import pandas as pd
import numpy as np

train = pd.read_csv('Data Analysis/Titanic/dataset/train.csv')
sns.boxplot(train['Age'])
plt.title("Box Plot antes da substituição")
plt.show()
q1 = train['Age'].quantile(0.25)
q3 = train['Age'].quantile(0.75)
iqr = q3-q1
Lower_tail = q1 - 1.5 * iqr
Upper_tail = q3 + 1.5 * iqr
m = np.mean(train['Age'])
for i in train['Age']:
    if i > Upper_tail or i < Lower_tail:
            train['Age'] = train['Age'].replace(i, m)
sns.boxplot(train['Age'])
plt.title("Box Plot após substituição - média")
plt.show()
```

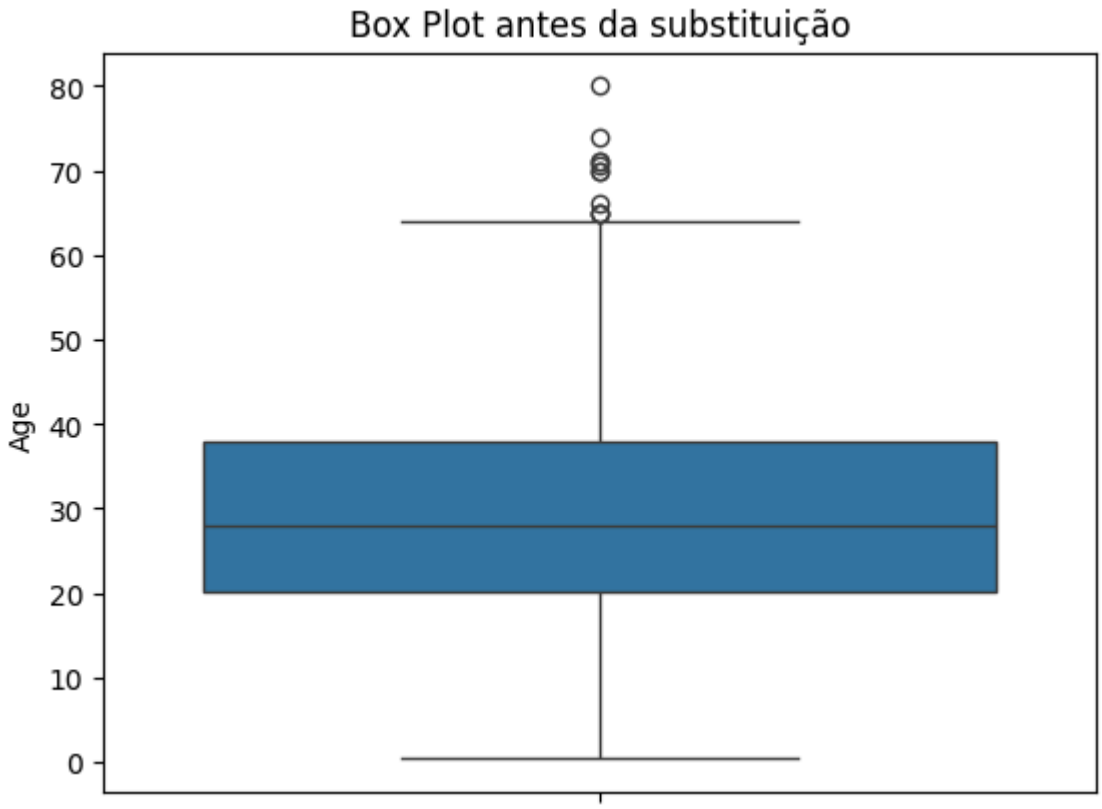


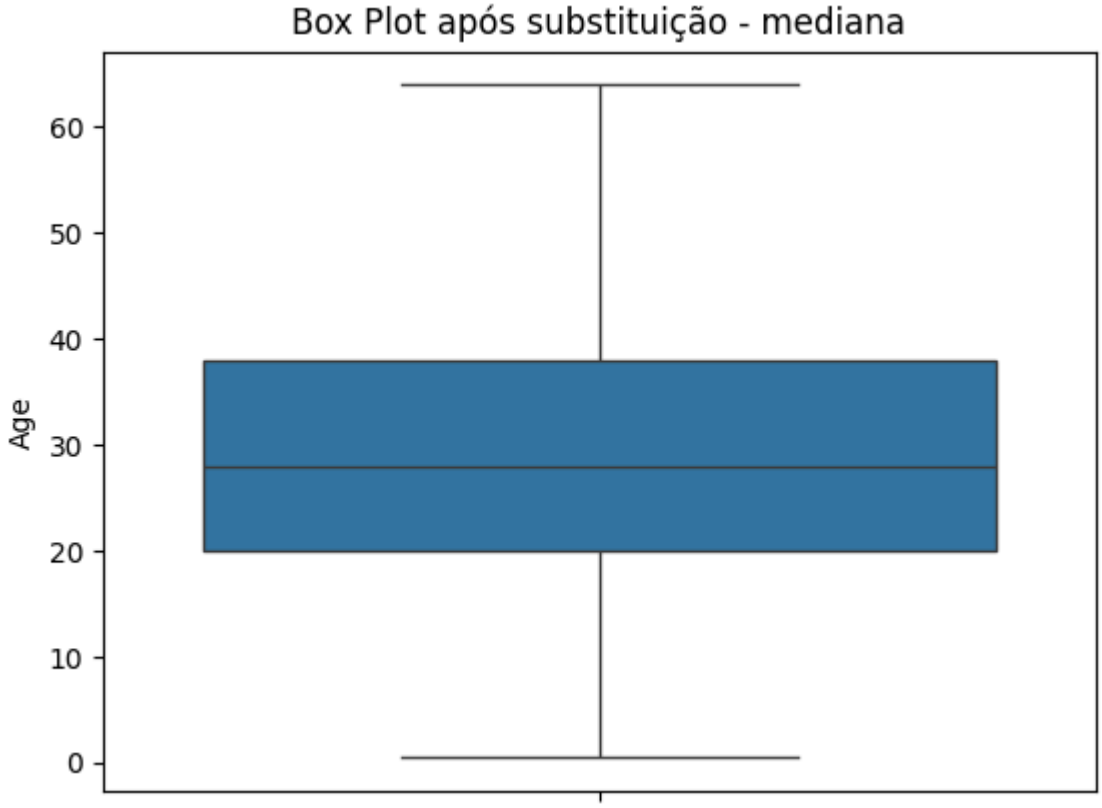


In [ ]:

```
#median imputation
import pandas as pd
import numpy as np

train = pd.read_csv('Data Analysis/Titanic/dataset/train.csv')
sns.boxplot(train['Age'])
plt.title("Box Plot antes da substituição")
plt.show()
q1 = train['Age'].quantile(0.25)
q3 = train['Age'].quantile(0.75)
iqr = q3-q1
```

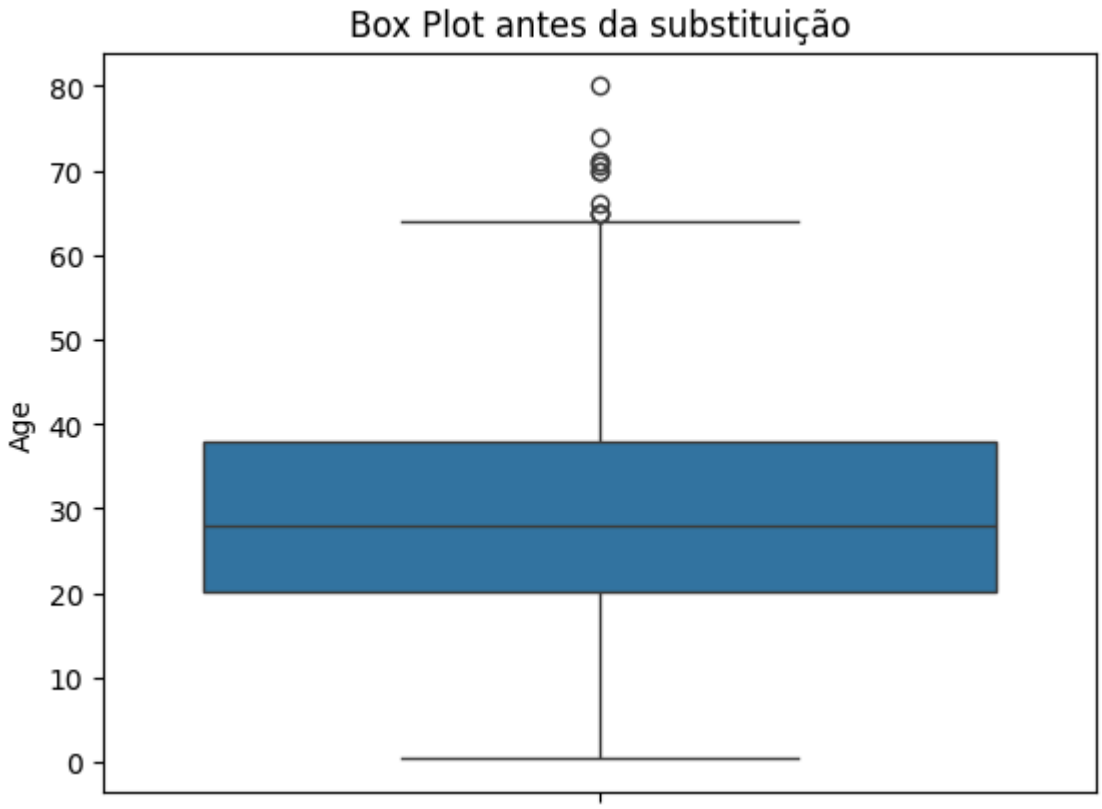
```
Lower_tail = q1 - 1.5 * iqr
Upper_tail = q3 + 1.5 * iqr
med = np.median(train['Age'])
for i in train['Age']:
    if i > Upper_tail or i < Lower_tail:
            train['Age'] = train['Age'].replace(i, med)
sns.boxplot(train['Age'])
plt.title("Box Plot após substituição - mediana")
plt.show()
```

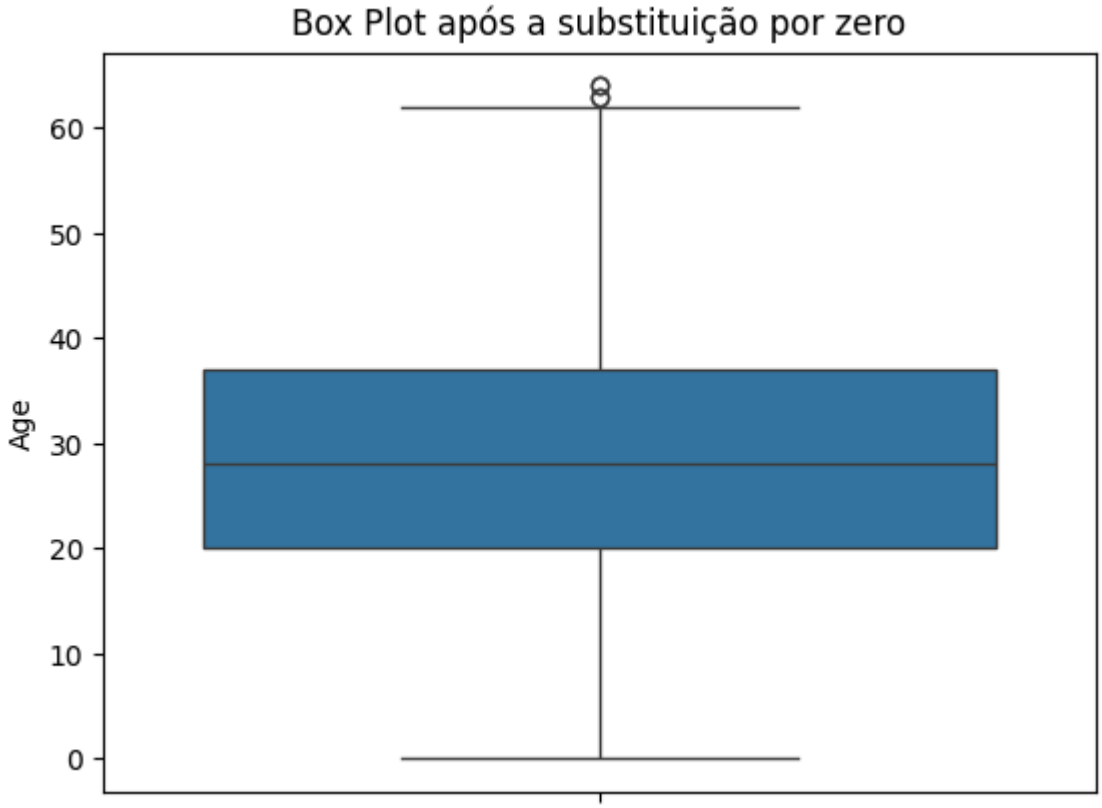




In [ ]:

```
#Zero value imputation
import pandas as pd
import numpy as np

train = pd.read_csv('Data Analysis/Titanic/dataset/train.csv')
sns.boxplot(train['Age'])
plt.title("Box Plot antes da substituição")
plt.show()
q1 = train['Age'].quantile(0.25)
q3 = train['Age'].quantile(0.75)
iqr = q3-q1
Lower_tail = q1 - 1.5 * iqr
Upper_tail = q3 + 1.5 * iqr
for i in train['Age']:
    if i > Upper_tail or i < Lower_tail:
            train['Age'] = train['Age'].replace(i, 0)
sns.boxplot(train['Age'])
plt.title("Box Plot após a substituição por zero")
plt.show()
```





TRATAMENTO SEPARADO

Se houver um número significativo de outliers e o conjunto de dados for pequeno, devemos tratá-los separadamente no modelo estatístico. Uma das abordagens é tratar ambos os grupos como dois grupos diferentes e construir modelos individuais para ambos os grupos e depois combinar os resultados. Mas esta técnica é trabalhosa caso o conjunto de dados seja grande.

Note que algumas das técnicas mencionadas podem nem sempre fornecer os melhores resultados. Portanto, tenha cuidado ao tentar detectar/substituir valores discrepantes.

REFERÊNCIAS

1. Finding outliers in dataset using python
2. Outlier Detection - Basics
3. Ways to Detect and Remove the Outliers
4. Four Techniques for Outlier Detection - KDnuggets
5. Outlier!!! The Silent Killer