About This Script:

Eeste script foi desenvolvido para calcular os quartis Q1 e Q3 para a coluna de preços e utiliza esses valores para categorizar os preços et classificar preços em 'Baixo', 'Médio' e 'Alto' utilizando quartis e visualiza os resultados em gráficos estilizados para Dark Mode.

O que sao Quartis e Quantis:

Quartis e quantis são medidas estatísticas que ajudam a dividir um conjunto de dados em partes iguais, mas têm significados ligeiramente diferentes.

Quartis:
- Dividem um conjunto de dados em quatro partes iguais.
- Existem três quartis principais:
  - Primeiro Quartil (Q1): Separa os 25% menores valores dos 75% maiores.
  - Segundo Quartil (Q2): É a mediana, dividindo o conjunto de dados ao meio.
  - Terceiro Quartil (Q3): Separa os 75% menores valores dos 25% maiores.

Quantis:
- São valores que dividem um conjunto de dados em partes iguais de forma mais geral.
- Incluem percentis (100 partes), decis (10 partes), e quartis (4 partes).

Diferença:

- Quartis são um tipo específico de quantil que divide os dados em quatro partes iguais.
- Quantis são mais gerais e podem dividir os dados em qualquer número de partes iguais.

Neste script, usamos quartis para classificar os preços:
- Baixo: Preço abaixo do Primeiro Quartil (P < Q1)
- Médio: Preço entre o Primeiro Quartil e o Terceiro Quartil (Q1 ≤ P ≤ Q3)
- Alto: Preço acima do Terceiro Quartil (P > Q3)

Passos do script:
1. Carregar os dados e verificar valores ausentes.
2. Calcular quartis e classificar preços.
3. Visualizar resultados com histogramas e boxplots, incluindo melhorias como normalização e análise estatística.
4. Salvar o DataFrame modificado em um novo arquivo CSV.
"""

- **Importações**: O código começa importando as bibliotecas necessárias (`pandas`, `seaborn`, `matplotlib`).
- **Quartis**: O código calcula os quartis Q1 e Q3 para a coluna de preços e utiliza esses valores para categorizar os preços.
- **Gráficos**: Inclui histogramas, gráficos de barras, boxplots e outros gráficos para visualizar a distribuição dos dados com um tema darkmode.
- **Quartis e Gráficos:** O código inclui o cálculo de quartis, histogramas para diferentes variáveis, gráficos de barras, boxplots, e uma visualização detalhada dos quartis na distribuição dos preços.

- **Modo Escuro:** Todos os gráficos são gerados no modo escuro (`dark_background`), proporcionando uma aparência sofisticada.

Esse código pode ser copiado e colado diretamente em seu ambiente Python para gerar gráficos elegantes com explicações claras em cada passo.

O código a seguior gera graphics representing the Quartile Calculations.including outros tipos de grapgicos que ajudam a entender a distribuição dos dados e como os quartis dividem a distribuição, also providing the quartile calculation visualizations.

- **Tema Dark**: Todos os gráficos foram configurados para usar o modo escuro para um visual mais elegante.
- **Visualização de Quartis**: Foram adicionados gráficos específicos para visualizar a distribuição dos preços e as classificações com base nos quartis,

im, o gráfico que aparece na imagem anexada, que mostra a distribuição normal com os quartis destacados (Q1, Q2, Q3), foi recriado no código que compartilhei.

O código a seguir dera gráfico com o tema dark e inclui os seguintes elementos:

- **Curva de Distribuição Normal**: Uma curva normal padrão centrada na média (mu = 0) com desvio padrão (sigma = 1).

- **Quartis**: Q1, Q2 (mediana), e Q3 são indicados com linhas verticais vermelha, preta e verde, respectivamente.
- **Áreas Sombreadas**: A área entre Q1 e Q3 foi preenchida com uma cor azul semitransparente para destacar visualmente o intervalo interquartil.
- **Legendas**: Adicionei rótulos para identificar Q1, Q2, e Q3.

Esse gráfico serve para ajudar a entender a distribuição dos dados e como os quartis dividem a distribuição. Está totalmente adaptado para o tema escuro para uma visualização mais elegante.

```
import pandas as pd                # Importa a biblioteca pandas para manipulação de dados

import seaborn as sns              # Importa a biblioteca seaborn para criar gráficos

import matplotlib.pyplot as plt    # Importa a biblioteca matplotlib para customização e exibição de gráficos

import numpy as np                 # Importa a biblioteca numpy para cálculos numéricos

import plotly.express as px


# Carregando o DataFrame

df = pd.read_csv('/mnt/data/dados_limpos.csv')  # Altera para o caminho correto do seu arquivo CSV
```

```
# Verificação de valores ausentes na coluna 'price'
if df['price'].isnull().sum() > 0:
    print(f"Atenção: Existem {df['price'].isnull().sum()} valores
ausentes na coluna 'price'.")
    df['price'] = df['price'].fillna(df['price'].median())  #
Preenchendo com a mediana


# Normalizando a coluna 'price'
df['price_normalized'] = (df['price'] - df['price'].min()) /
(df['price'].max() - df['price'].min())




#

# Calculando os quartis da coluna 'price'
Q1 = df['price'].quantile(0.25)  # Calcula o primeiro quartil
(Q1), que é o valor abaixo do qual 25% dos dados estão

Q3 = df['price'].quantile(0.75)  # Calcula o terceiro quartil
(Q3), que é o valor abaixo do qual 75% dos dados estão


# Criando a coluna 'Categoria_Preco' e classificando
os preços
df['Categoria_Preco'] = 'Médio'  # Inicializando com 'Médio'
df.loc[df['price'] < Q1, 'Categoria_Preco'] = 'Baixo'  #
Preços abaixo do Primeiro Quartil
```

df.loc[df['price'] > Q3, 'Categoria_Preco'] = 'Alto' # Preços acima do Terceiro Quartil

 # Classificando os preços

df['Categoria_Preco'] = pd.cut(df['price'], bins=[-float('inf'), q1, q3, float('inf')], labels=['Baixo', 'Médio', 'Alto']) # Cria uma

 # Exibindo as primeiras linhas do DataFrame
print(df.head())

 # Análise estatística básica da coluna 'price'
mean_price = df['price'].mean()
median_price = df['price'].median()
std_price = df['price'].std()
print(f"Média do Preço: {mean_price:.2f}")
print(f"Mediana do Preço: {median_price:.2f}")
print(f"Desvio Padrão do Preço: {std_price:.2f}")

 # Visualizando os quartis graficamente

plt.figure(figsize=(10, 6)) # Define o tamanho da figura do gráfico

sns.histplot(df['price'], kde=True, edgecolor='white', color='skyblue') # Cria um histograma com densidade (kde) para os preços

plt.axvline(q1, color='red', linestyle='--', label='Q1') # Adiciona uma linha vertical representando Q1 (primeiro quartil)

plt.axvline(q3, color='green', linestyle='--', label='Q3') # Adiciona uma linha vertical representando Q3 (terceiro quartil)

plt.legend() # Adiciona uma legenda para identificar as linhas Q1 e Q3

plt.title('Distribuição dos Preços com Quartis') # Define o título do gráfico

plt.xlabel('Preço') # Define o rótulo do eixo X

plt.ylabel('Frequência') # Define o rótulo do eixo Y

plt.style.use('dark_background') # Aplica o tema dark para o gráfico

plt.show() # Exibe o gráfico

 # Configuração do estilo para Dark Mode
plt.style.use('dark_background')

 # Visualização: Boxplot de Preço por Categoria

plt.figure(figsize=(10, 6))

plt.figure(figsize=(10, 6)) # Define o tamanho da figura do gráfico

sns.countplot(x='Categoria_Preco', data=df, palette='viridis', edgecolor='black') # Cria um gráfico de barras para contar as categorias de preço

plt.title('Distribuição das Categorias de Preço', fontsize=16, weight='bold') # Define o título do gráfico com formatação de texto

plt.xlabel('Categoria de Preço', fontsize=12) # Define o rótulo do eixo X com formatação de texto

plt.ylabel('Contagem', fontsize=12) # Define o rótulo do eixo Y com formatação de texto

plt.xticks(fontsize=10, weight='bold') # Formata os rótulos do eixo X

plt.yticks(fontsize=10, weight='bold') # Formata os rótulos do eixo Y

plt.style.use('dark_background') # Aplica o tema dark para o gráfico

plt.show() # Exibe o gráfico

```
plt.figure(figsize=(10, 6))
sns.boxplot(x='Categoria_Preco', y='price', data=df,
palette='coolwarm')
plt.title('Boxplot de Preço por Categoria')
plt.xlabel('Categoria de Preço')
plt.ylabel('Preço')
plt.show()
```

 # Histograma básico para 'curb-weight'

plt.figure(figsize=(10, 6)) # Define o tamanho da figura do gráfico

df['curb-weight'].hist(edgecolor='k', color='skyblue') # Cria um histograma básico para 'curb-weight' com bordas pretas

plt.title('Distribuição do Peso em Curva (Curb Weight)') # Define o título do gráfico

plt.xlabel('Peso em Curva') # Define o rótulo do eixo X

plt.ylabel('Frequência') # Define o rótulo do eixo Y

plt.style.use('dark_background') # Aplica o tema dark para o gráfico

plt.show() # Exibe o gráfico

Visualização: Histograma de Preços com linha de densidade

plt.figure(figsize=(10, 6))
sns.histplot(df['price'], kde=True, color='cyan', bins=30, edgecolor='white')
plt.title('Histograma de Preços com Densidade')
plt.xlabel('Preço')
plt.ylabel('Frequência')
plt.axvline(Q1, color='red', linestyle='dashed', linewidth=2, label='Q1')

```
plt.axvline(Q3, color='green', linestyle='dashed', linewidth=2, label='Q3')
plt.legend()
plt.show()
```

 # Histograma com 3 bins

```
plt.figure(figsize=(10, 6))  # Define o tamanho da figura do gráfico

df['curb-weight'].hist(edgecolor='k', bins=3, color='skyblue')  # Cria um histograma com 3 bins para 'curb-weight' com bordas pretas

plt.title('Distribuição do Peso em Curva com 3 Bins')  # Define o título do gráfico

plt.xlabel('Peso em Curva')  # Define o rótulo do eixo X

plt.ylabel('Frequência')  # Define o rótulo do eixo Y

plt.style.use('dark_background')  # Aplica o tema dark para o gráfico

plt.show()  # Exibe o gráfico
```

 # Visualização: Gráfico de violino para mostrar a distribuição dos preços em cada categoria

```
plt.figure(figsize=(10, 6))
sns.violinplot(x='Categoria_Preco', y='price', data=df, palette='coolwarm')
plt.title('Distribuição de Preços por Categoria')
plt.xlabel('Categoria de Preço')
plt.ylabel('Preço')
plt.show()
```

 # Classificação de 'curb-weight' em 3 categorias

 # Cria uma nova coluna 'Curb_Weight_Category' no DataFrame, categorizando 'curb-weight' em 'leve', 'médio' e 'pesado'

```
df['Curb_Weight_Category'] = pd.cut(df['curb-weight'], 3, labels=['leve', 'médio', 'pesado'])
```

 # Visualização interativa com Plotly

```
fig = px.histogram(df, x='price', color='Categoria_Preco', nbins=30,
                   title='Histograma Interativo de Preços por Categoria')
fig.show()
```

 # Visualização do conceito de quartis (gráfico semelhante ao da imagem fornecida)

plt.figure(figsize=(12, 8)) # Define o tamanho da figura do gráfico

mu = 0 # Média

sigma = 1 # Desvio padrão

x = np.linspace(mu - 3*sigma, mu + 3*sigma, 100) # Define os valores no eixo X

plt.plot(x, 1/(sigma * np.sqrt(2 * np.pi)) * np.exp(-(x - mu)**2 / (2 * sigma**2)), color='brown') # Desenha a curva de distribuição normal

plt.fill_between(x, 0, 1/(sigma * np.sqrt(2 * np.pi)) * np.exp(-(x - mu)**2 / (2 * sigma**2)), where=(x > q1) & (x < q3), color='blue', alpha=0.5)

 # Preenche a área entre Q1 e Q3 com azul

plt.axvline(mu, color='black', linestyle='-', label='Q2 (Mediana)') # Adiciona uma linha vertical para a mediana (Q2)

plt.axvline(mu - sigma, color='red', linestyle='--', label='Q1 (Primeiro Quartil)') # Adiciona uma linha vertical para Q1

plt.axvline(mu + sigma, color='green', linestyle='--', label='Q3 (Terceiro Quartil)') # Adiciona uma linha vertical para Q3

plt.text(mu - sigma, 0.02, 'Q1', horizontalalignment='center', color='red', fontsize=12) # Adiciona o texto 'Q1' na posição correspondente

plt.text(mu + sigma, 0.02, 'Q3', horizontalalignment='center', color='green', fontsize=12) # Adiciona o texto 'Q3' na posição correspondente

plt.text(mu, 0.02, 'Q2', horizontalalignment='center', color='black', fontsize=12) # Adiciona o texto 'Q2' na posição correspondente

plt.title('Entendendo os Quartis na Distribuição Normal', fontsize=16) # Define o título do gráfico

plt.xlabel('Distribuição Normal') # Define o rótulo do eixo X

plt.ylabel('Densidade') # Define o rótulo do eixo Y

plt.legend() # Adiciona uma legenda para as linhas verticais

plt.style.use('dark_background') # Aplica o tema dark para o gráfico

plt.show() # Exibe o gráfico

 # Visualização interativa com Plotly

```
fig = px.histogram(df, x='price', color='Categoria_Preco', nbins=30,
                   title='Histograma Interativo de Preços por Categoria')
fig.show()
```

 # Salvando o DataFrame modificado em um novo arquivo CSV

```
df.to_csv('dados_classificados.csv', index=False)
print("DataFrame classificado salvo em 'dados_classificados.csv'.")
```