

Programa de Pós-graduação em Computação Aplicada – PPCA (UnB)

Prova Final de AEDI - 02/02/2025

Aluno: Alexandre Quirino de Melo

Matrícula: 241130411

#####

#####

Questão 1: Esta questão aborda a aplicação prática de um problema de Ciência de Dados utilizando Regressão Linear. O objetivo é prever preços de imóveis com base em dados reais da região de King County, nos Estados Unidos. A base de dados utilizada é a Previsão de Vendas de Imóveis em King County (EUA). Siga os passos abaixo para desenvolver sua solução:

Instruções:

1. Análise Descritiva dos Dados (20%)
- Realize uma análise inicial da base de dados.
 - Inclua estatísticas descritivas (média, mediana, desvio padrão, etc.) e gráficos relevantes (distribuições, correlações, etc.).
2. Construção do Modelo de Regressão Linear (30%)
- Construa um modelo de Regressão Linear para prever os preços dos imóveis.
 - Apresente os coeficientes do modelo, R2 e outras métricas de avaliação.
3. Interpretação dos Resultados (10%)
- Explique os resultados obtidos pelo modelo, destacando o impacto de cada variável nas previsões e explicações do fenômeno.
 - Verifique se os pressupostos da Regressão Linear (linearidade, homocedasticidade, normalidade dos resíduos, etc.) foram atendidos.
4. Ajustes no Modelo (30%)
- Identifique possíveis problemas nos pressupostos do modelo.
 - Apresente soluções para corrigir esses problemas, como transformações de variáveis ou ajustes no modelo.
 - Reavalie o desempenho do modelo ajustado.
5. Tomada de Decisão (10%)
- Com base no modelo final, explique como os resultados podem ser aplicados em um contexto de negócios.
 - Forneça exemplos de decisões estratégicas que poderiam ser tomadas com base nas previsões.


#####

1. Análise Descritiva dos Dados

#####

```
# Bibliotecas
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import statsmodels.api as sm
from google.colab import files
from google.colab import files

# Fazer upload do arquivo (kc_house_data.csv)
uploaded = files.upload()
file_name = 'kc_house_data.csv'
```

 Escolher arquivos

Nenhum arquivo escolhido Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable

```
# Carregar os dados
data = pd.read_csv('kc_house_data.csv')

# Definindo a semente para garantir a reprodutibilidade
SEED = 65
random.seed(SEED)
np.random.seed(SEED)

# 1.1 - Análise descritiva básica
print(data.describe())
```

	count	2.161300e+04	2.161300e+04	21613.000000	21613.000000	21613.000000	\
	mean	4.580302e+09	5.400881e+05	3.370842	2.114757	2079.899736	
	std	2.876566e+09	3.671272e+05	0.930062	0.770163	918.440897	
	min	1.000102e+06	7.500000e+04	0.000000	0.000000	290.000000	
	25%	2.123049e+09	3.219500e+05	3.000000	1.750000	1427.000000	
	50%	3.904930e+09	4.500000e+05	3.000000	2.250000	1910.000000	
	75%	7.308900e+09	6.450000e+05	4.000000	2.500000	2550.000000	
	max	9.900000e+09	7.700000e+06	33.000000	8.000000	13540.000000	
		sqft_lot	floors	waterfront	view	condition	\
	count	2.161300e+04	21613.000000	21613.000000	21613.000000	21613.000000	
	mean	1.510697e+04	1.494309	0.007542	0.234303	3.409430	
	std	4.142051e+04	0.539989	0.086517	0.766318	0.650743	
	min	5.200000e+02	1.000000	0.000000	0.000000	1.000000	
	25%	5.040000e+03	1.000000	0.000000	0.000000	3.000000	
	50%	7.618000e+03	1.500000	0.000000	0.000000	3.000000	
	75%	1.068800e+04	2.000000	0.000000	0.000000	4.000000	
	max	1.651359e+06	3.500000	1.000000	4.000000	5.000000	
		grade	sqft_above	sqft_basement	yr_built	yr_renovated	\
	count	21613.000000	21613.000000	21613.000000	21613.000000	21613.000000	
	mean	7.656873	1788.390691	291.509045	1971.005136	84.402258	
	std	1.175459	828.090978	442.575043	29.373411	401.679240	
	min	1.000000	290.000000	0.000000	1900.000000	0.000000	
	25%	7.000000	1190.000000	0.000000	1951.000000	0.000000	
	50%	7.000000	1560.000000	0.000000	1975.000000	0.000000	
	75%	8.000000	2210.000000	560.000000	1997.000000	0.000000	
	max	13.000000	9410.000000	4820.000000	2015.000000	2015.000000	
		zipcode	lat	long	sqft_living15	sqft_lot15	
	count	21613.000000	21613.000000	21613.000000	21613.000000	21613.000000	
	mean	98077.939805	47.560053	-122.213896	1986.552492	12768.455652	
	std	53.505026	0.138564	0.140828	685.391304	27304.179631	
	min	98001.000000	47.155900	-122.519000	399.000000	651.000000	
	25%	98033.000000	47.471000	-122.328000	1490.000000	5100.000000	
	50%	98065.000000	47.571800	-122.230000	1840.000000	7620.000000	
	75%	98118.000000	47.678000	-122.125000	2360.000000	10083.000000	
	max	98199.000000	47.777600	-121.315000	6210.000000	871200.000000	

✓ Interpretação geral dos dados

Amostra:

- O conjunto de dados possui 21.613 registros (observações).
- Os valores foram analisados com base em estatísticas descritivas como média, desvio padrão, mínimo, percentis e máximo.

Preço (price):

- **Média:** O preço médio das casas é \$540.088.
- **Desvio padrão:** Há uma grande variação nos preços (\$367.127).
- **Mínimo e máximo:** Os preços variam de 75.000a7.700.000, indicando uma ampla gama de propriedades, desde acessíveis até luxuosas.
- **Mediana:** A mediana (\$450.000) é menor que a média, sugerindo uma assimetria positiva (maior concentração de preços abaixo da média).

Quartos (bedrooms):

- **Média:** A maioria das casas tem cerca de 3 quartos.
- **Máximo:** Há uma casa com 33 quartos, o que é um outlier óbvio.
- **Percentis:** 50% das casas têm 3 quartos, e 75% têm até 4 quartos.

Banheiros (bathrooms):

- **Média:** 2,11 banheiros por casa.
- **Máximo:** 8 banheiros, também indicando propriedades de luxo.
- **Percentis:** A maior parte das casas tem entre 1,75 e 2,5 banheiros.

Área construída (sqft_living):

- **Média:** A área construída média é de 2.079 pés quadrados.
- **Desvio padrão:** Variabilidade significativa (~918 pés quadrados).
- **Percentis:** A maioria das casas tem entre 1.427 e 2.550 pés quadrados.

Área do lote (sqft_lot):

- **Média:** Lotes com uma área média de 15.107 pés quadrados.
- **Máximo:** Um lote com 1.651.359 pés quadrados, o que é um outlier.

Características adicionais**Floors (andares):**

- A maioria das casas tem entre 1 e 2 andares, com uma média de 1,49. Casas com até 3,5 andares estão presentes, mas são menos comuns.

Waterfront (frente para água):

- Apenas 0,75% das casas estão à beira da água, indicando exclusividade.

View (vista):

- 75% das casas têm vista classificada como 0 (sem vista relevante).
- Algumas casas têm vistas classificadas em até 4 (máxima qualidade).

Condition (condição):

- A condição média das casas é classificada como 3,4 (em uma escala de 1 a 5).
- 75% têm condição classificada como 4 ou inferior.

Grade (qualidade de construção e design):

- Média de 7,65, o que sugere qualidade acima da média.
- Variação entre 1 (mínima) e 13 (máxima).

Ano de construção (yr_built):

- **Média:** 1971.
- **Mediana:** 1975.
- Indica uma maioria de casas construídas no final do século XX.

Ano de renovação (yr_renovated):

A maioria das casas não foi renovada (mediana = 0). Algumas foram renovadas até 2015.

Localização (latitude e longitude):

- A dispersão dos dados de latitude e longitude reflete a distribuição geográfica das propriedades.

Outliers e considerações**Outliers óbvios:**

- Número de quartos (33 quartos).
- Lotes extremamente grandes (>1 milhão de pés quadrados).
- Casas muito grandes (>13.000 pés quadrados de área construída).
- Casas extremamente caras (\$7.700.000).

Breve Conclusão:

- Os outliers podem distorcer análises estatísticas. É importante considerar sua remoção ou ajuste dependendo do objetivo do estudo.
- Variáveis como "waterfront" e "view" têm baixa incidência de valores elevados, mas podem ser importantes para prever o preço das casas.

Próximos passos:

- Explorar a relação entre as variáveis, especialmente o impacto de "sqft_living", "waterfront", "view" e "grade" no preço.
- Tratar os outliers para evitar viés nos modelos preditivos.
- Considerar transformações em variáveis como preço para lidar com assimetria positiva.

```
# 1.2 - Verificar valores ausentes
print(data.isnull().sum())
```

```
# Obs: 0 resultado abaixo indica que não há valores ausentes em nenhuma das
#      colunas do dataset. Isso significa que todas as linhas possuem dados
#      completos para todas as variáveis presentes.
```

```

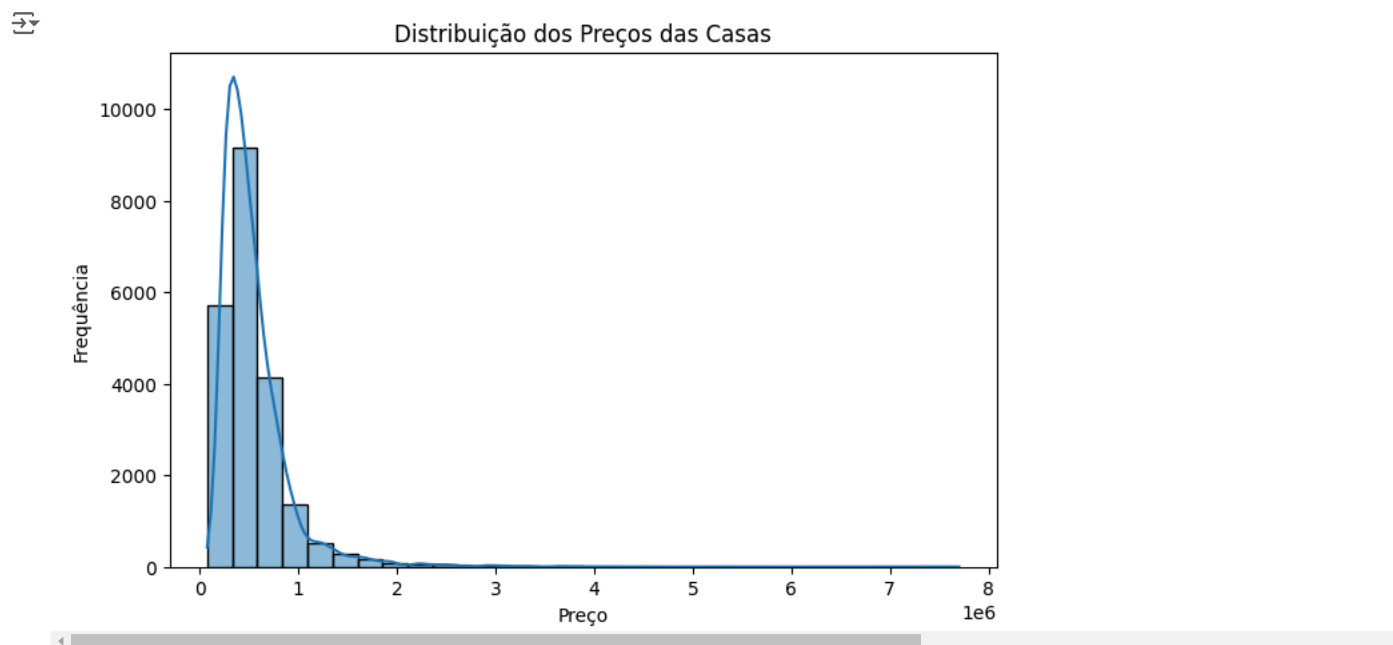
id          0
date        0
price       0
bedrooms    0
bathrooms   0
sqft_living 0
sqft_lot    0
floors      0
waterfront  0
view        0
condition   0
grade       0
sqft_above  0
sqft_basement 0
yr_built    0
yr_renovated 0
zipcode     0
lat         0
long        0
sqft_living15 0
sqft_lot15  0
dtype: int64

```

```

# 1.3 - Histograma do preço das casas
plt.figure(figsize=(8, 5))
sns.histplot(data['price'], kde=True, bins=30)
plt.title('Distribuição dos Preços das Casas')
plt.xlabel('Preço')
plt.ylabel('Frequência')
plt.show()

```



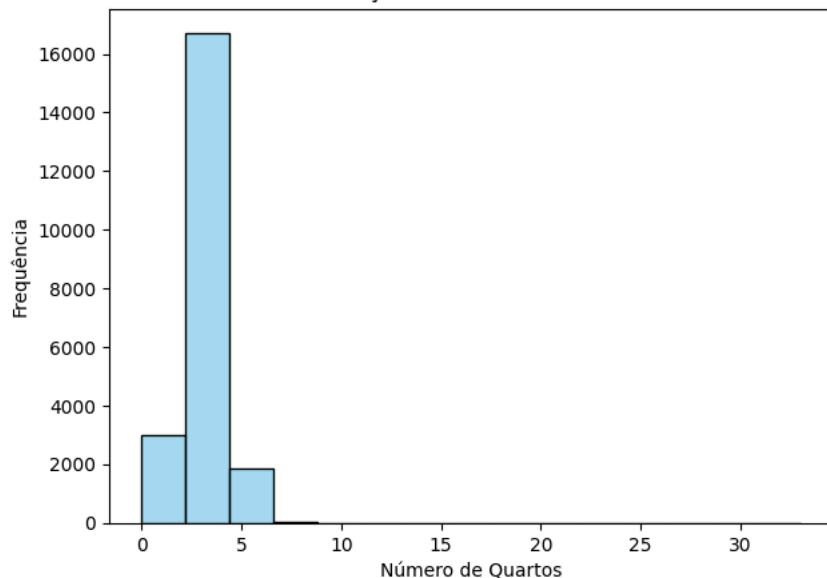
```

# 1.4 - Histograma do número de quartos (bedrooms)
plt.figure(figsize=(7, 5))
sns.histplot(data['bedrooms'], bins=15, kde=False, color='skyblue')
plt.title('Distribuição do Número de Quartos', fontsize=12)
plt.xlabel('Número de Quartos', fontsize=10)
plt.ylabel('Frequência', fontsize=10)
plt.show()

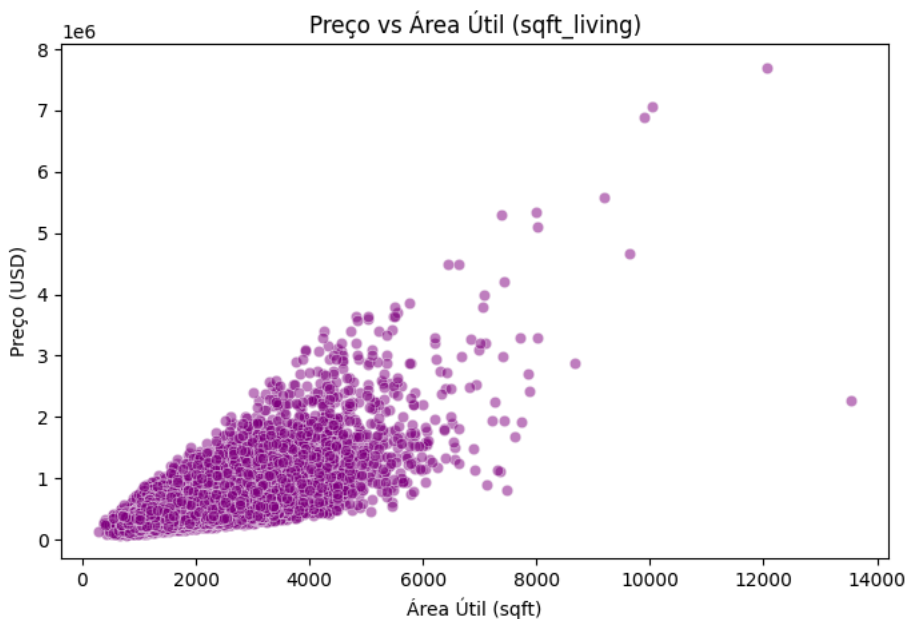
```



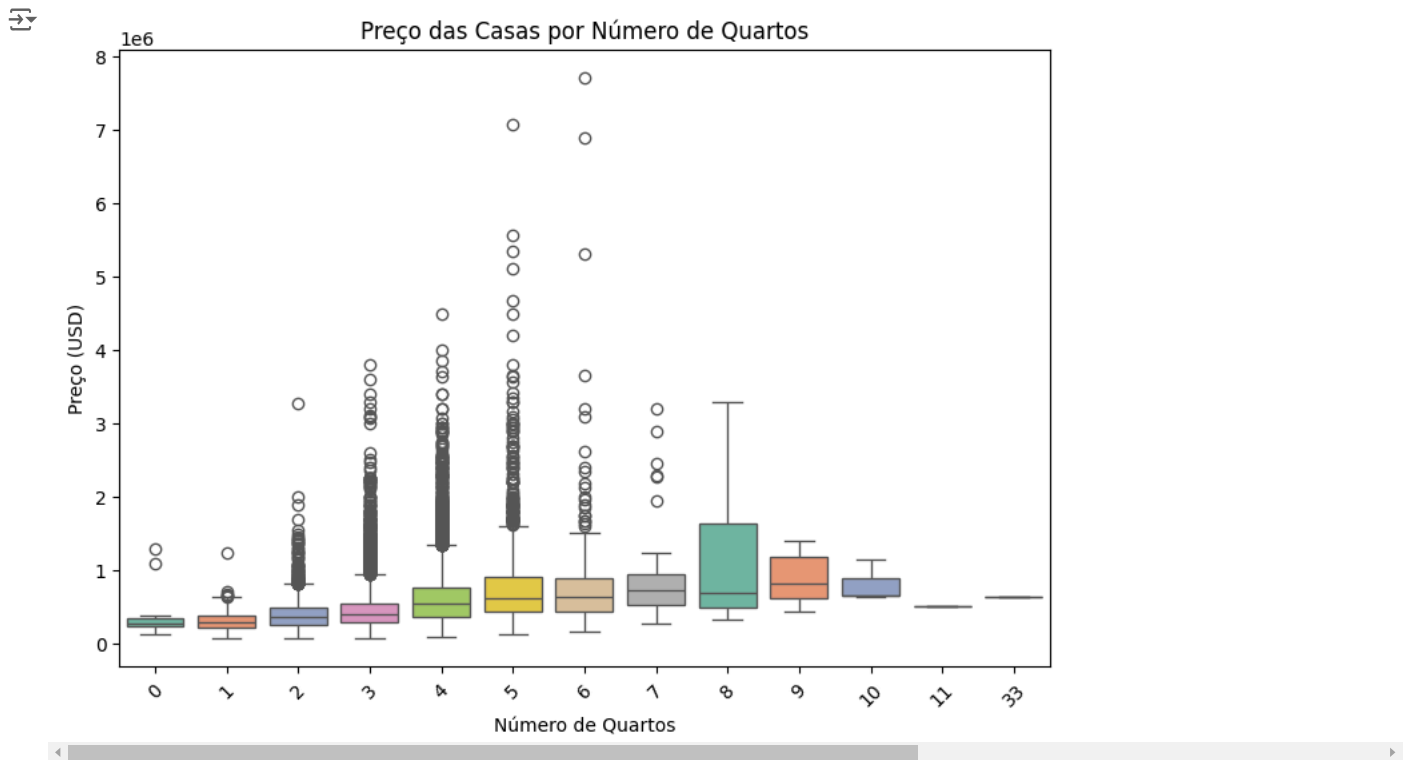
Distribuição do Número de Quartos



```
# 1.5 - Gráfico de dispersão: Preço (price) vs. Área útil (sqft_living)
plt.figure(figsize=(8, 5))
sns.scatterplot(x=data['sqft_living'], y=data['price'], alpha=0.5, color='purple')
plt.title('Preço vs Área Útil (sqft_living)', fontsize=12)
plt.xlabel('Área Útil (sqft)', fontsize=10)
plt.ylabel('Preço (USD)', fontsize=10)
plt.show()
```



```
# 1.6 - Boxplot: Preço das casas por número de quartos (bedrooms)
plt.figure(figsize=(9, 6))
sns.boxplot(x='bedrooms', y='price', data=data, hue='bedrooms', palette='Set2', legend=False)
plt.title('Preço das Casas por Número de Quartos', fontsize=12)
plt.xlabel('Número de Quartos', fontsize=10)
plt.ylabel('Preço (USD)', fontsize=10)
plt.xticks(rotation=45)
plt.show()
```

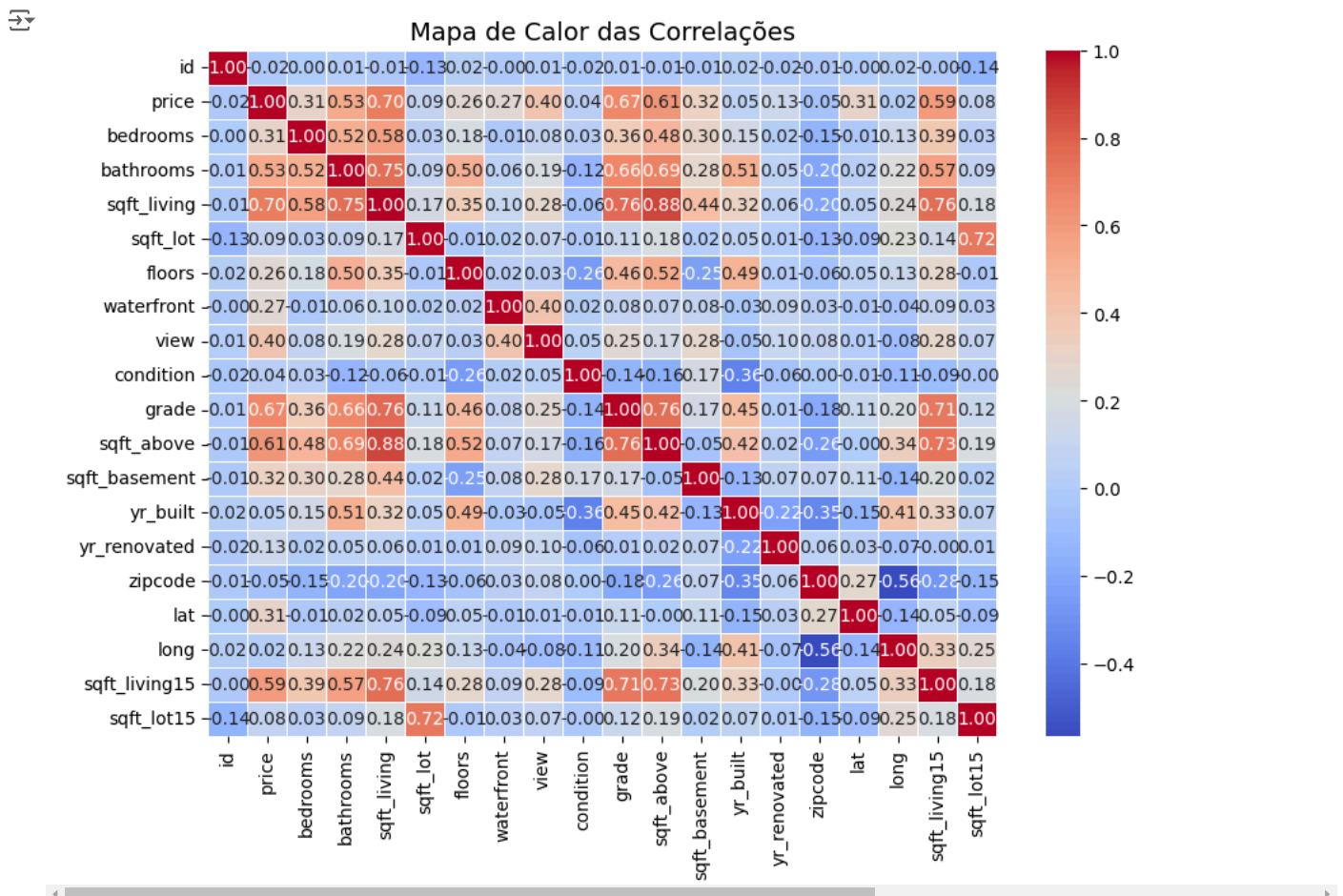


1.7 - Mapa de calor das correlações entre as variáveis numéricas

```
# Filtrar apenas colunas numéricas
numeric_columns = data.select_dtypes(include=['float64', 'int64'])
```

```
# Calcular a matriz de correlação
correlation_matrix = numeric_columns.corr()
```

```
# Gerar o mapa de calor
plt.figure(figsize=(10, 7))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
plt.title('Mapa de Calor das Correlações', fontsize=14)
plt.show()
```



```
# 1.8 - Gráfico de barras: Condição das casas (condition)
plt.figure(figsize=(8, 5))
sns.countplot(x=data['condition'], palette='viridis', hue=data['condition'], dodge=False)
plt.title('Contagem de Propriedades por Condição', fontsize=12)
plt.xlabel('Condição', fontsize=10)
plt.ylabel('Contagem', fontsize=10)
plt.show()
```

```
# Observações para Condição:
```

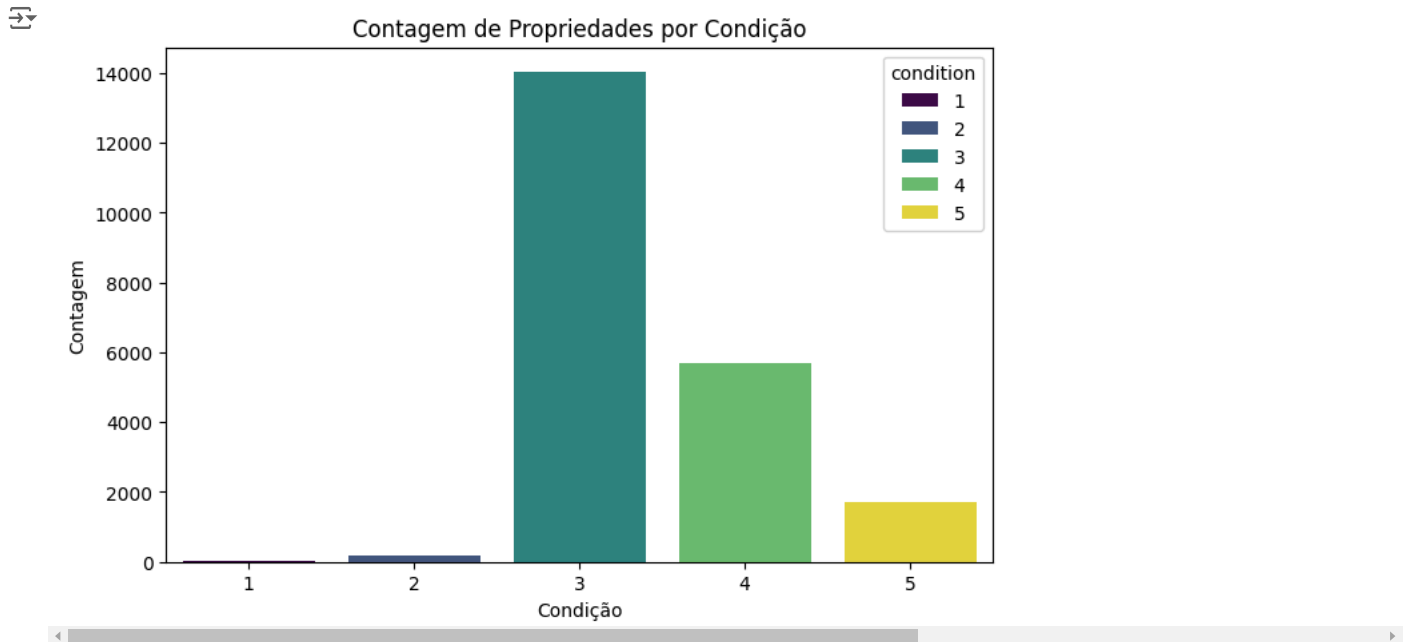
```
# 1: "Muito ruim",
```

```
# 2: "Ruim",
```

```
# 3: "Regular",
```

```
# 4: "Boa",
```

```
# 5: "Excelente"
```



```
# 1.9 - Distribuição geográfica: Latitude (lat) vs Longitude (long)
plt.figure(figsize=(10, 5))
```

```
# Criar scatter plot com cores representando o preço
```

```
scatter = plt.scatter(
    x=data['long'],
    y=data['lat'],
    c=data['price'],
    cmap='viridis',
    alpha=0.6
)
```

```
# Adicionar título e rótulos
```

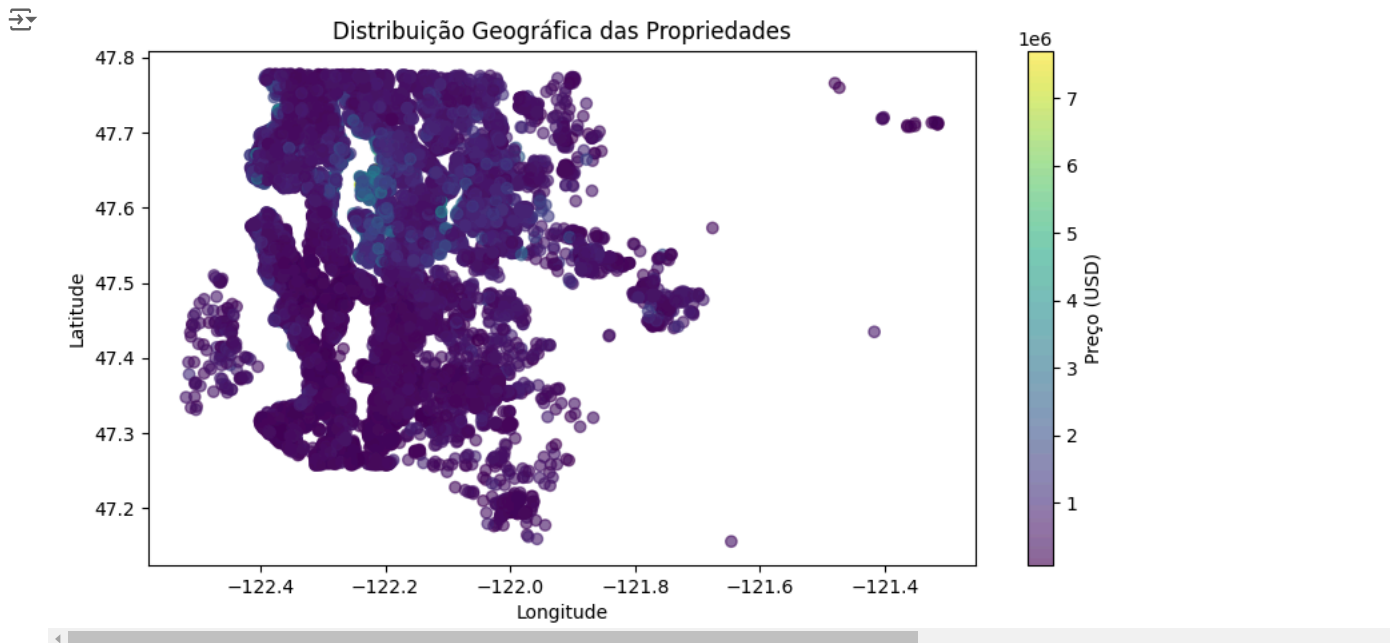
```
plt.title('Distribuição Geográfica das Propriedades', fontsize=12)
plt.xlabel('Longitude', fontsize=10)
plt.ylabel('Latitude', fontsize=10)
```

```
# Adicionar barra de cores
```

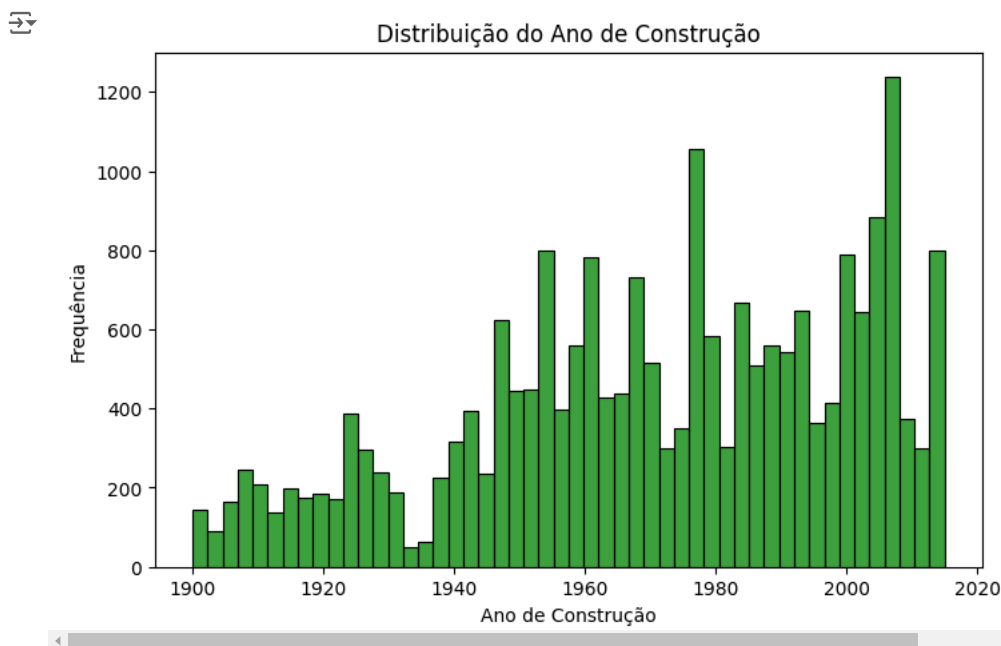
```
plt.colorbar(scatter, label='Preço (USD)')
```

```
# Mostrar o gráfico
```

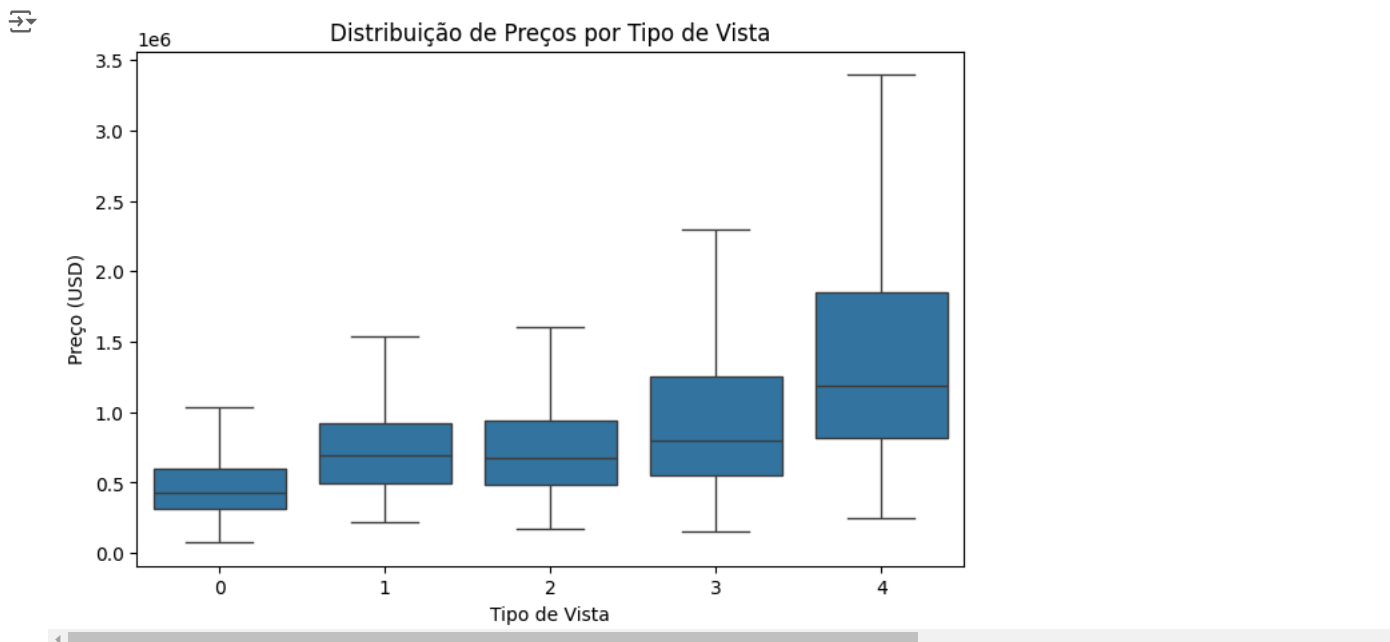
```
plt.show()
```



```
# 1.10 - Histograma: Ano de construção (yr_built)
plt.figure(figsize=(8, 5))
sns.histplot(data['yr_built'], bins=50, kde=False, color='green')
plt.title('Distribuição do Ano de Construção', fontsize=12)
plt.xlabel('Ano de Construção', fontsize=10)
plt.ylabel('Frequência', fontsize=10)
plt.show()
```

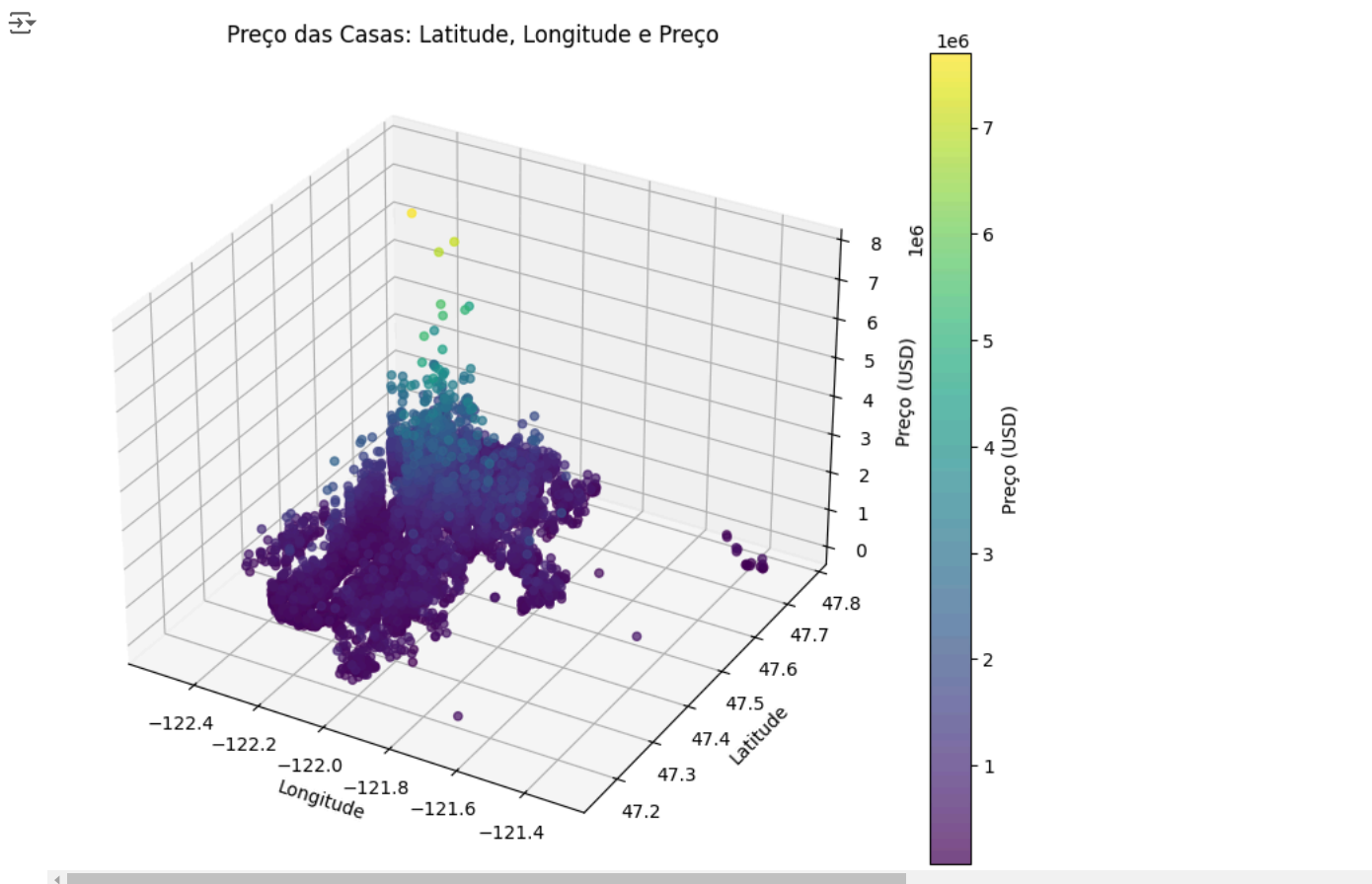


```
# 1.11 - Boxplot: Preço por vista (view)
plt.figure(figsize=(8, 5))
sns.boxplot(x=data['view'], y=data['price'], showfliers=False)
plt.title('Distribuição de Preços por Tipo de Vista', fontsize=12)
plt.xlabel('Tipo de Vista', fontsize=10)
plt.ylabel('Preço (USD)', fontsize=10)
plt.show()
```

1.12 - Gráfico de dispersão 3D: Preço (price), Latitude (lat) e Longitude (long)
 from mpl_toolkits.mplot3d import Axes3D

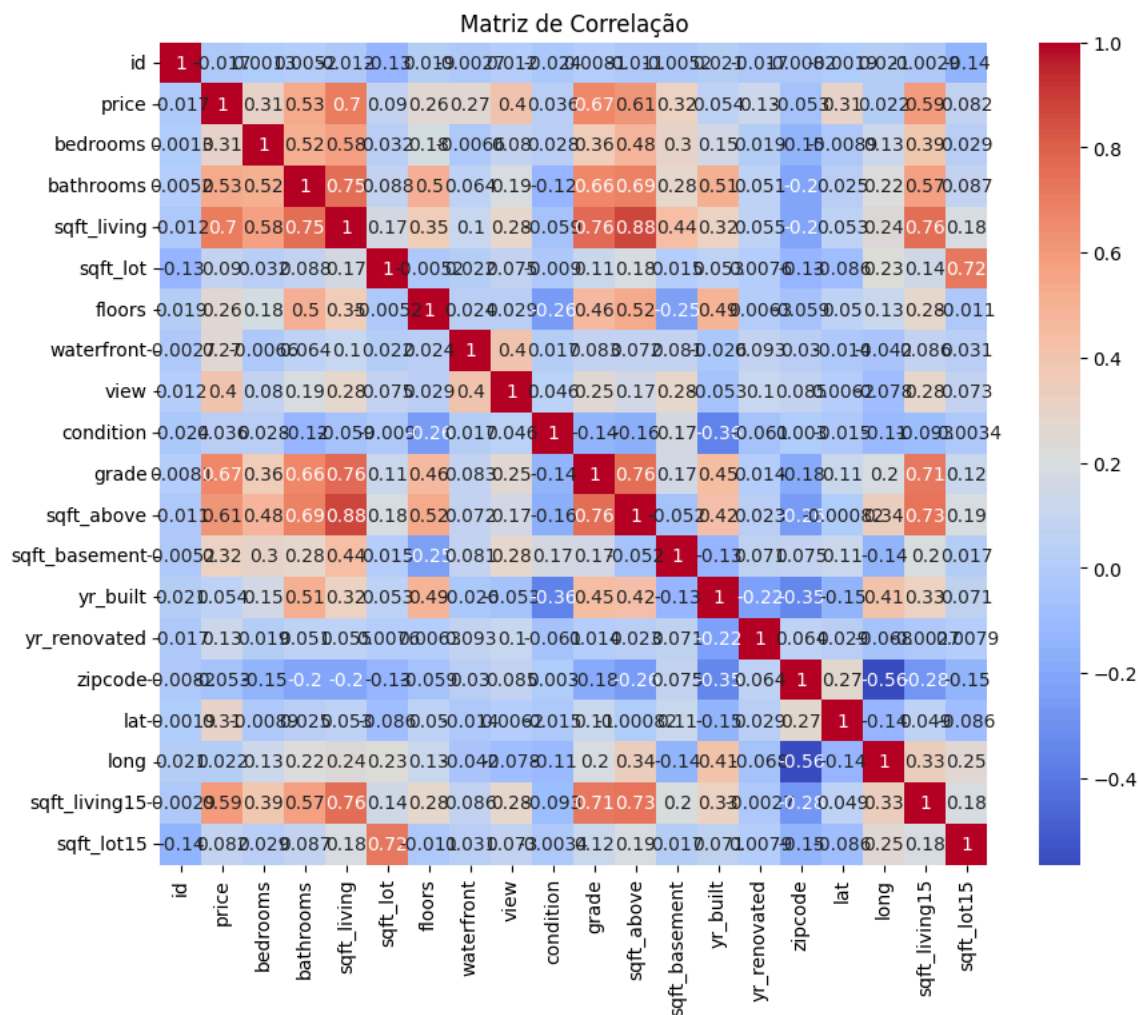
```
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')
sc = ax.scatter(data['long'], data['lat'], data['price'], c=data['price'], cmap='viridis', alpha=0.7)
plt.colorbar(sc, label='Preço (USD)')
ax.set_title('Preço das Casas: Latitude, Longitude e Preço', fontsize=12)
ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')
ax.set_zlabel('Preço (USD)')
plt.show()
```



```
# 1.13 - Gráfico colunas numéricas para a matriz de correlação
numerical_data = data.select_dtypes(include=[np.number])
corr_matrix = numerical_data.corr()

plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
```

```
plt.title('Matriz de Correlação')
plt.show()
```



```
#####
#### 2. Construção do Modelo de Regressão Linear
#####
```

```
# Selecionar variáveis independentes e dependente
X = data[['sqft_living', 'bedrooms', 'bathrooms', 'floors', 'condition', 'grade', 'sqft_above', 'sqft_basement', 'yr_built',
y = data['price']
```

```
# Dividir os dados em treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

```
# Ajustar o modelo de regressão linear
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```



LinearRegression

LinearRegression()

```
# 2.1 - Prever no conjunto de teste
y_pred = regressor.predict(X_test)
print(y_pred)
```



```
[ 517414.12990427  701703.84615009 1160682.3811053 ... 287876.58162408
 892441.21502576  497309.86312464]
```

```
# Resultados sumarizados
print(f'Coeficientes: {regressor.coef_}')
print(f'Intercepto: {regressor.intercept_}')
print(f'Erro Quadrático Médio (MSE): {mean_squared_error(y_test, y_pred)}')
print(f'R²: {r2_score(y_test, y_pred)}')
```



```
Coeficientes: [ 1.27764471e+02 -4.72454125e+04  4.69761670e+04  3.01282325e+04
 1.66140485e+04  1.28536123e+05  5.13305748e+01  7.64338964e+01
-3.81197169e+03  2.34772137e+01]
```

Intercepto: 6645546.795774356
Erro Quadrático Médio (MSE): 56324980805.193954
R²: 0.6098473781804925

Interpretando os resultados acima:

1. Coeficientes: Esses são os coeficientes do modelo de regressão, que indicam a relação entre as variáveis independentes (features) e a variável dependente (preço das casas). Cada valor representa o quanto o preço das casas varia com o aumento de uma unidade de cada variável independente, mantendo as outras constantes.

Por exemplo:

- 1.27764471e+02** significa que, para cada aumento de 1 unidade em bedrooms (número de quartos), o preço da casa aumenta em aproximadamente 127,76 unidades de moeda, assumindo que as outras variáveis permaneçam constantes.
- 4.72454125e+04** indica que, para cada aumento de 1 unidade em sqft_living (área construída), o preço da casa diminui em aproximadamente 47.245 unidades de moeda, assumindo as outras variáveis constantes.
- E assim por diante para as outras variáveis.

2. Intercepto: O intercepto (ou constante) indica o valor estimado da variável dependente (preço das casas) quando todas as variáveis independentes são iguais a zero. Neste caso, o preço estimado da casa seria **6.645.546**, o que representa um preço básico de casa, antes de levar em conta as características como o número de quartos, área, etc.

3. Erro Quadrático Médio (MSE): O MSE é uma medida da média dos quadrados dos erros, ou seja, a diferença média entre os valores previstos pelo modelo (y_{pred}) e os valores reais (y_{test}). Quanto menor o MSE, melhor é o modelo. No seu caso, o MSE é **56.324.980.805,19**, o que indica um erro considerável. Isso pode sugerir que o modelo tem dificuldade em prever com precisão os preços das casas, especialmente em comparação com os valores reais.

4. R² (R-quadrado): O R² é a proporção da variabilidade total da variável dependente que é explicada pelo modelo. Ele varia entre 0 e 1, onde:

- 0** significa que o modelo não explica nenhuma variabilidade dos dados.
- 1** significa que o modelo explica toda a variabilidade.

Neste caso, o valor de **0.6098 (aproximadamente 61%)** indica que o modelo consegue explicar cerca de 61% da variabilidade no preço das casas. Isso é relativamente bom, mas ainda há uma quantidade considerável de variabilidade que o modelo não consegue capturar.

Breve Resumo:

- Coeficientes indicam como cada característica (como o número de quartos, área, etc.) afeta o preço das casas.
- O **intercepto** fornece o preço base quando todas as características são zero.
- O **MSE** é relativamente alto, sugerindo que o modelo pode não ser extremamente preciso.
- O **R²** de aproximadamente 0.61 sugere que o modelo tem uma capacidade razoável de explicar a variação no preço das casas, mas há espaço para melhorias.

Com base nesses resultados, o modelo pode ser ajustado ou melhorado, por exemplo, considerando a inclusão de novas variáveis, a aplicação de técnicas de regularização, ou até a utilização de outros tipos de modelos mais complexos (como modelos de árvores de decisão ou redes neurais) para obter previsões mais precisas.

```
# 2.2 - Usar statsmodels para um resumo mais detalhado
X_train_sm = sm.add_constant(X_train) # adicionar constante para o termo de intercepto
model = sm.OLS(y_train, X_train_sm).fit()
print(model.summary())
```

yr_built	-3811.9717	86.689	-43.973	0.000	-3981.893	-3642.050
yr_renovated	23.4772	4.864	4.827	0.000	13.944	33.011

Omnibus:	12619.006	Durbin-Watson:	2.005
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1092067.975
Skew:	3.501	Prob(JB):	0.00
Kurtosis:	44.029	Cond. No.	1.26e+16

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

[2] The smallest eigenvalue is 1.18e-21. This might indicate that there are strong multicollinearity problems or that the design matrix is singular.

✓ Interpretação dos resultados acima

Aqui está uma interpretação detalhada dos resultados da regressão OLS (Mínimos Quadrados Ordinários):

1. R-quadrado (R^2) e R-quadrado ajustado (Adj. R^2):

- **R^2 :** 0.621 — Isso significa que 62.1% da variabilidade no preço das casas é explicada pelas variáveis independentes no modelo. Em outras palavras, o modelo consegue capturar mais de 60% da variação do preço.
- **R^2 ajustado:** 0.621 — O R^2 ajustado leva em consideração o número de variáveis no modelo e penaliza a inclusão de variáveis que não contribuem significativamente. Como o R^2 ajustado é muito próximo do R^2 , isso indica que as variáveis no modelo são relevantes.

2. F-estatístico e p-valor:

- **F-estatístico:** 2753 — Este valor mede a qualidade do modelo. Um valor alto de F sugere que o modelo é significativamente melhor do que um modelo sem variáveis explicativas.
- **Prob (F-estatístico):** 0.00 — Este p-valor extremamente baixo (próximo de 0) indica que o modelo como um todo é estatisticamente significativo, ou seja, pelo menos uma das variáveis independentes é significativamente relacionada ao preço.

3. Coeficientes:

Os coeficientes indicam o impacto de cada variável independente sobre o preço das casas, assumindo que todas as outras variáveis permanecem constantes.

- **Constante (intercepto):** 6.646e+06 — Quando todas as variáveis independentes são zero, o preço da casa seria aproximadamente 6.646.000. Esse valor é o ponto de partida para a previsão do preço.
- **sqft_living:** 127.76 — Para cada aumento de 1 metro quadrado na área útil da casa, o preço da casa aumenta em 127,76 unidades de moeda, mantendo as outras variáveis constantes. Este coeficiente tem um impacto positivo muito forte no preço.
- **bedrooms:** -47.25 — Cada quarto adicional diminui o preço da casa em 47.250 unidades de moeda, assumindo que as outras variáveis sejam constantes. Isso pode parecer contra-intuitivo, mas pode refletir que casas com muitos quartos não necessariamente têm preços mais altos (por exemplo, casas maiores, com mais quartos, podem ter uma distribuição de área menos eficiente).
- **bathrooms:** 46.98 — Cada banheiro adicional aumenta o preço em 46.980 unidades de moeda. Mais banheiros geralmente indicam uma casa mais confortável, o que aumenta seu valor.
- **floors:** 30.13 — Cada andar adicional aumenta o preço da casa em 30.130 unidades de moeda. Isso sugere que casas com mais andares tendem a ser mais caras.
- **condition:** 16.61 — Melhorias na condição da casa (por exemplo, renovação ou conservação) aumentam o preço em 16.610 unidades de moeda.
- **grade:** 128.5 — A qualidade do acabamento da casa (ou "nota de qualidade") tem um grande impacto, com um aumento de 128.500 unidades de moeda no preço da casa para cada unidade adicional na nota de qualidade.
- **sqft_above:** 51.33 — Cada aumento de 1 metro quadrado na área acima do solo (sem contar o porão) aumenta o preço em 51,33 unidades de moeda.
- **sqft_basement:** 76.43 — Cada aumento de 1 metro quadrado de área no porão aumenta o preço em 76,43 unidades de moeda. Isso sugere que áreas subterrâneas bem desenvolvidas podem agregar valor à casa.
- **yr_built:** -3811.97 — Casas mais antigas tendem a ter preços mais baixos. O preço diminui em 3.812 unidades de moeda por cada ano a mais de construção da casa. Este coeficiente negativo pode ser uma indicação de que, em média, casas mais antigas precisam de mais manutenção ou não têm as características desejadas pelos compradores.
- **yr_renovated:** 23.48 — Cada ano adicional de renovação da casa aumenta o preço em 23.480 unidades de moeda. Renovar uma casa tende a aumentar seu valor, o que é refletido nesse coeficiente positivo.

4. Erros padrão (std err):

Estes valores indicam a precisão dos coeficientes estimados. Quanto menores os erros padrão, mais confiáveis são as estimativas dos coeficientes.

- Por exemplo, o erro padrão de `sqft_living` é 2.76, o que é relativamente pequeno em relação ao coeficiente de 127.76, indicando que a estimativa do coeficiente é muito precisa.

5. Estatísticas t e p-valores:

- **t e P>|t|** testam a hipótese nula de que o coeficiente é igual a zero (sem efeito).
- Para todos os coeficientes, o p-valor é muito baixo (0.000), o que significa que todas as variáveis são estatisticamente significativas no modelo.

6. Diagnósticos do modelo:

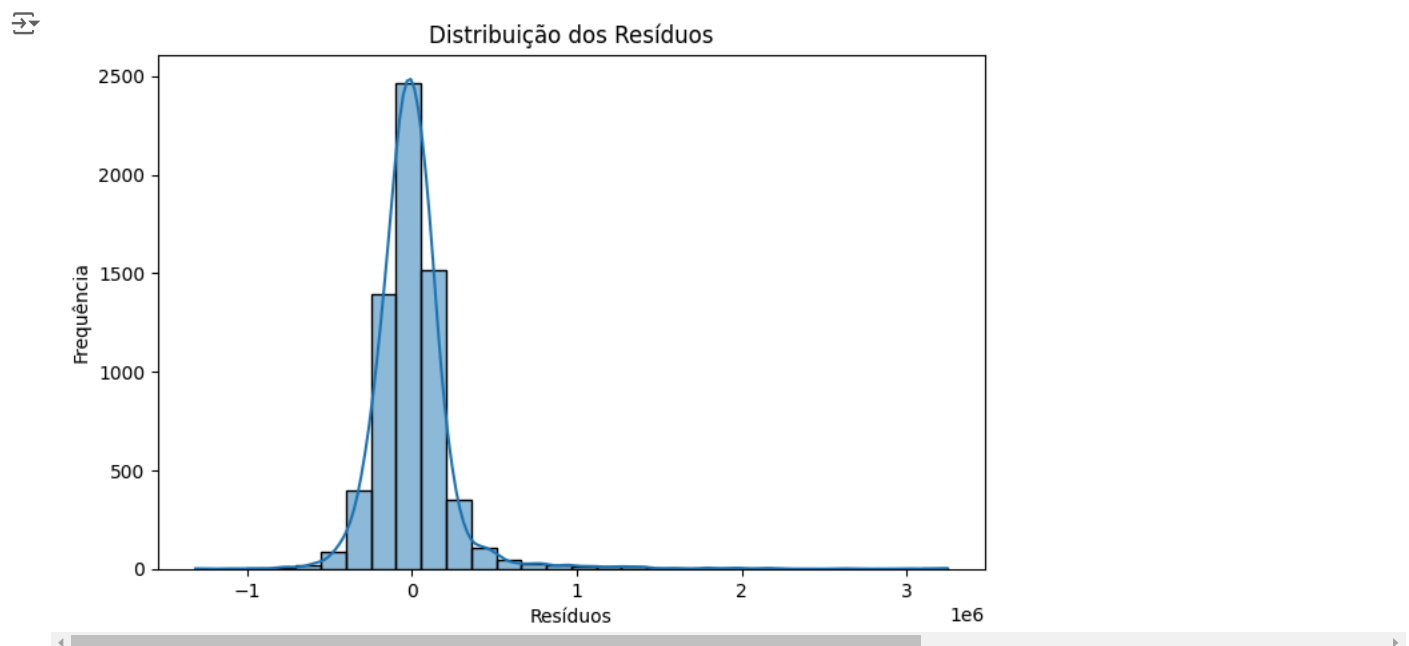
- **Omnibus:** 12619.006 e Prob(Omnibus): 0.000 — O teste Omnibus verifica se os resíduos seguem uma distribuição normal. O valor de p muito baixo sugere que os resíduos não seguem exatamente uma distribuição normal, o que pode indicar que o modelo não está capturando todos os padrões dos dados.
- **Durbin-Watson:** 2.005 — Esse teste verifica a autocorrelação dos resíduos. O valor de 2.0 sugere que não há autocorrelação significativa entre os resíduos, o que é um bom sinal.
- **Jarque-Bera (JB):** 1092067.975 e Prob(JB): 0.000 — O teste Jarque-Bera também verifica a normalidade dos resíduos. O p-valor baixo sugere que os resíduos não são normalmente distribuídos.
- **Cond. No:** 1.26e+16 — Um número de condição alto (acima de 30) pode indicar problemas de multicolinearidade (quando as variáveis independentes estão altamente correlacionadas). Um número tão alto sugere que pode haver multicolinearidade no modelo, o que pode afetar a precisão das estimativas.

Breve Conclusão:

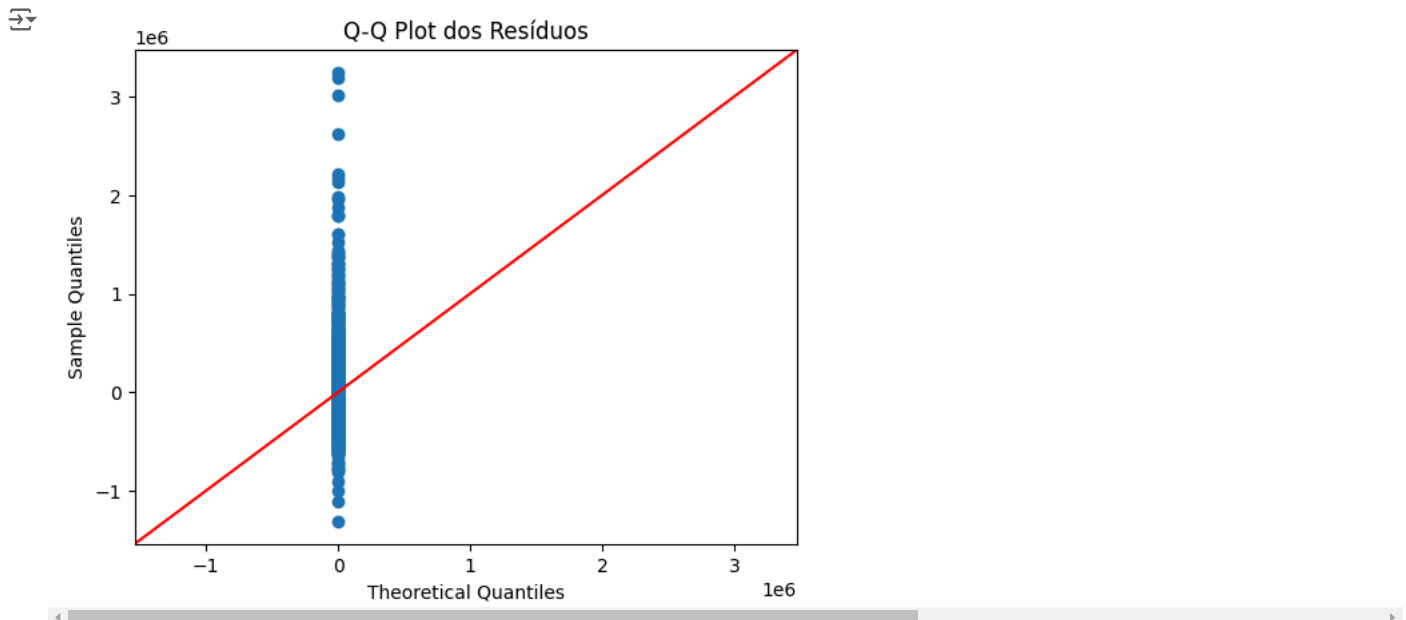
- O modelo é geral e as variáveis independentes são estatisticamente significativas para prever o preço das casas.
- R^2 de 62% indica que o modelo explica uma boa parte da variação no preço, mas há espaço para melhorias.
- O modelo pode sofrer com problemas de multicolinearidade e a distribuição não normal dos resíduos, o que pode indicar que ele não está capturando completamente as características dos dados.

```
#####
#### 3. Interpretação dos Resultados
#####
```

```
# 3.1 - Verificação da normalidade dos resíduos
plt.figure(figsize=(8, 5))
sns.histplot(residuals, kde=True, bins=30)
plt.title('Distribuição dos Resíduos')
plt.xlabel('Resíduos')
plt.ylabel('Frequência')
plt.show()
```



```
# 3.2 - Q-Q plot para verificar a normalidade dos resíduos
sm.qqplot(residuals, line='45')
plt.title('Q-Q Plot dos Resíduos')
plt.show()
```



✓ Conclusões e Fenômeno

Com base nos coeficientes e no R^2 , podemos concluir que:

- Variáveis como a área útil (sqft_living), a qualidade da construção (grade), e o estado geral da casa (condition) têm um impacto significativo no preço da casa.
- A idade da casa (yr_built) tem uma relação negativa com o preço, o que é esperado, já que casas mais antigas tendem a ser menos valorizadas, especialmente se não foram renovadas.
- Variáveis como o número de quartos (bedrooms) têm um impacto menor ou até negativo, sugerindo que o número de quartos pode não ser um dos fatores mais relevantes para determinar o preço, especialmente se a casa não for bem distribuída.
- O modelo explica uma boa parte da variabilidade do preço (R^2 de 62.1%), mas ainda há espaço para melhorias, possivelmente com a inclusão de outras variáveis (como localização e características do bairro) ou o uso de modelos mais complexos.

3.3 - Para verificar se os pressupostos da regressão linear foram atendidos, precisamos realizar
uma série de verificações. Os pressupostos típicos da regressão linear incluem:

Linearidade: A relação entre as variáveis independentes e dependentes deve ser linear.
Independência dos resíduos: Os resíduos devem ser independentes uns dos outros.
Homoscedasticidade: A variância dos resíduos deve ser constante para todos os níveis da variável independente.
Normalidade dos resíduos: Os resíduos devem seguir uma distribuição normal.
Ausência de multicolinearidade: As variáveis independentes não devem estar altamente correlacionadas entre si.

Carregar os dados e separar variáveis

```
X = data[['sqft_living', 'bedrooms', 'bathrooms', 'floors', 'condition', 'grade', 'sqft_above', 'sqft_basement', 'yr_built',
y = data['price']
```

Dividir os dados em treino e teste

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Ajustar o modelo de regressão linear

```
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Previsões no conjunto de treino

```
y_train_pred = regressor.predict(X_train)
```

Calcular os resíduos (diferença entre o valor real e o valor previsto)

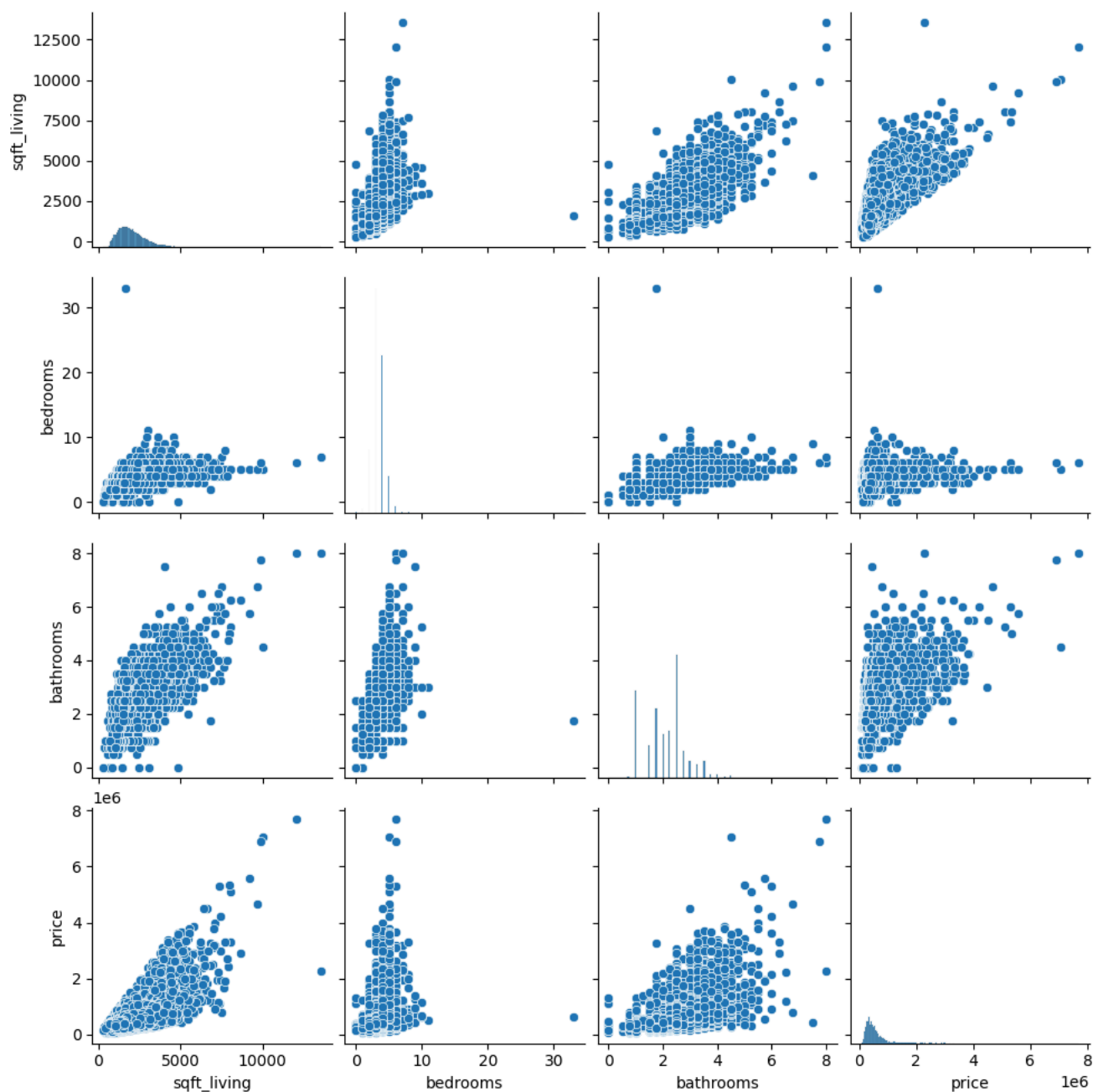
```
residuos = y_train - y_train_pred
```

3.4 - Verificação de linearidade: gráfico de dispersão entre as variáveis independentes e dependente

```
plt.figure(figsize=(9, 5))
sns.pairplot(data[['sqft_living', 'bedrooms', 'bathrooms', 'price']], kind='scatter')
plt.suptitle("Gráficos de Dispersão para Verificação de Linearidade", y=1.02)
plt.show()
```

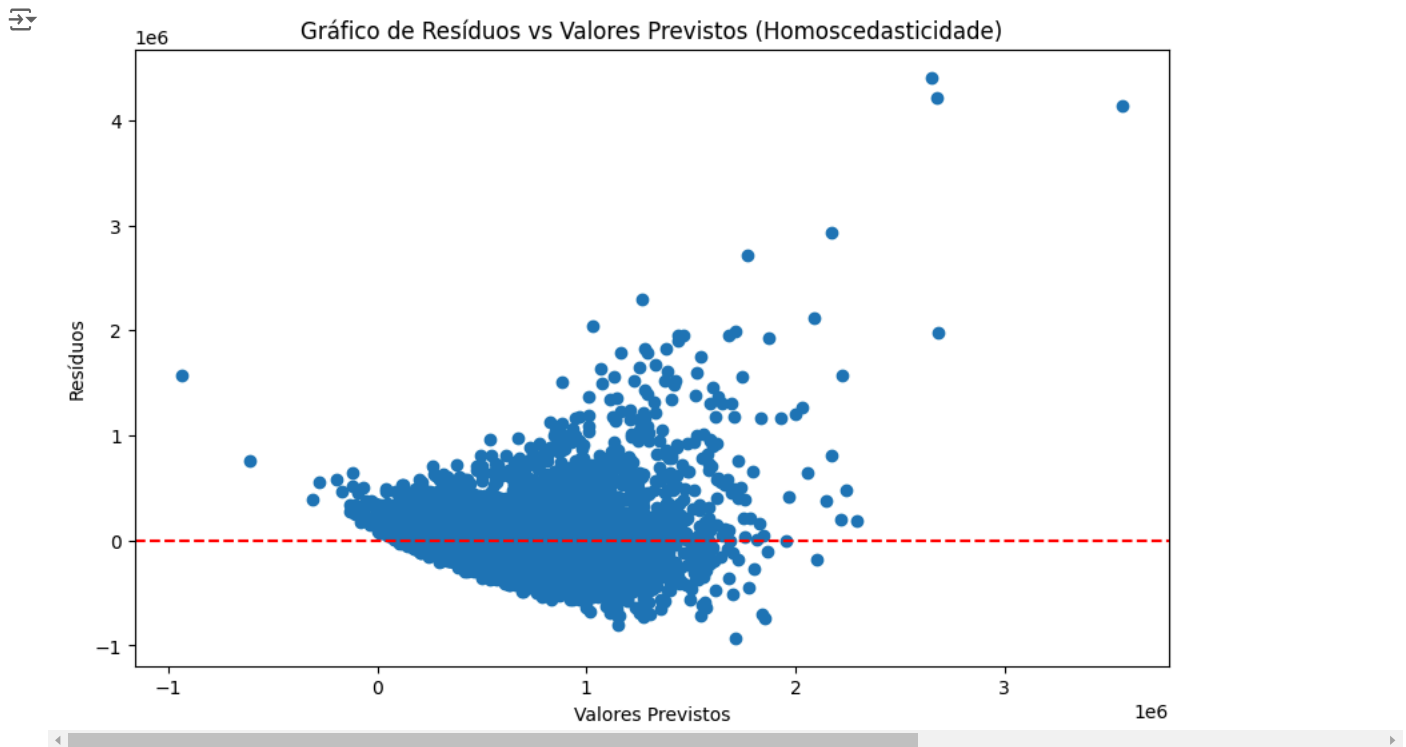
<Figure size 900x500 with 0 Axes>

Gráficos de Dispersão para Verificação de Linearidade



3.5 - Verificação de homocedasticidade: gráfico de resíduos vs valores previstos

```
plt.figure(figsize=(10, 6))
plt.scatter(y_train_pred, residuos)
plt.axhline(0, color='red', linestyle='--')
plt.title("Gráfico de Resíduos vs Valores Previstos (Homoscedasticidade)")
plt.xlabel("Valores Previstos")
plt.ylabel("Resíduos")
plt.show()
```



```
# 3.6 - Teste de normalidade: Teste de Shapiro-Wilk
import scipy.stats as stats
```

```
stat, p_value = stats.shapiro(residuos)
print(f"Resultado do Teste de Shapiro-Wilk: estatística={stat:.3f}, p-valor={p_value:.3f}")
if p_value > 0.05:
    print("Os resíduos seguem uma distribuição normal.")
else:
    print("Os resíduos não seguem uma distribuição normal.")
```

```
Resultado do Teste de Shapiro-Wilk: estatística=0.812, p-valor=0.000
Os resíduos não seguem uma distribuição normal.
/usr/local/lib/python3.10/dist-packages/scipy/stats/_axis_nan_policy.py:531: UserWarning: scipy.stats.shapiro: For N > 5
res = hypotest_fun_out(*samples, **kws)
```

✓ Interpretação dos resultados acima

O resultado do teste de Shapiro-Wilk indicou que os resíduos não seguem uma distribuição normal:

- **Estatística do teste (W):** 0.812
- **p-valor:** 0.000

Estatística do teste: Quanto mais próxima de 1 for a estatística do teste, mais provável é que os resíduos sigam uma distribuição normal. O valor obtido (0.812) é consideravelmente distante de 1, sugerindo que os resíduos não seguem uma distribuição normal.

p-valor: O p-valor é extremamente baixo (0.000), indicando que a hipótese nula de normalidade dos resíduos deve ser rejeitada. Ou seja, não podemos concluir que os resíduos seguem uma distribuição normal.

Sobre o Aviso que está no resultado:

- O aviso gerado pelo `scipy.stats.shapiro` indica que para amostras maiores que 5000 (como no seu caso, com 15129 observações), o p-valor pode não ser preciso. Isso ocorre devido à limitação do teste Shapiro-Wilk para grandes amostras.

O que isso significa para o seu modelo de regressão linear?

- A regressão linear pressupõe que os resíduos (diferença entre os valores reais e as previsões) sigam uma distribuição normal. Como o teste de Shapiro-Wilk indica que os resíduos não seguem essa distribuição, isso sugere que o pressuposto de normalidade não é atendido, o que pode impactar a validade de alguns testes de significância estatística do modelo (como os testes t para os coeficientes).

```
# 3.7 - Teste de independência dos resíduos: Teste de Durbin-Watson
dw_stat = sm.stats.durbin_watson(residuos)
print(f"Estatística de Durbin-Watson: {dw_stat:.3f}")
if 1.5 < dw_stat < 2.5:
    print("Os resíduos parecem ser independentes.")
else:
    print("Os resíduos podem estar correlacionados (violação da independência).")
```


↗ Estatística de Durbin-Watson: 2.005
Os resíduos parecem ser independentes.

```
# 3.8 - Verificação de multicolinearidade: VIF (Variance Inflation Factor)
from statsmodels.stats.outliers_influence import variance_inflation_factor
X_train_sm = sm.add_constant(X_train)
vif_data = pd.DataFrame()
vif_data['Variável'] = X_train_sm.columns
vif_data['VIF'] = [variance_inflation_factor(X_train_sm.values, i) for i in range(X_train_sm.shape[1])]

print(vif_data)
```

	Variável	VIF
0	const	8759.109853
1	sqft_living	inf
2	bedrooms	1.598190
3	bathrooms	3.337794
4	floors	1.883431
5	condition	1.220369
6	grade	2.932508
7	sqft_above	inf
8	sqft_basement	inf
9	yr_built	1.978628
10	yr_renovated	1.139549

/usr/local/lib/python3.10/dist-packages/statsmodels/stats/outliers_influence.py:197: RuntimeWarning: divide by zero encountered in scalar divide
vif = 1. / (1. - r_squared_i)

Interpretação dos valores do VIF

Os resultados fornecidos correspondem aos valores do Variance Inflation Factor (**VIF**) para cada variável em um modelo de regressão linear. O VIF é utilizado para verificar a multicolinearidade entre as variáveis independentes do modelo. Ele quantifica o quanto a variância de um coeficiente de regressão é inflacionada devido à correlação com outras variáveis independentes.

Constante (const):

- VIF: 8759.11
- O VIF muito alto para a constante pode ser um reflexo do fato de que a constante é fortemente influenciada pela presença de outras variáveis no modelo. No entanto, o VIF da constante não é usualmente uma preocupação, pois é uma interceptação fixa.

sqft_living:

- VIF: inf (infinito)
- O VIF para sqft_living é infinito, o que sugere que há multicolinearidade perfeita com outras variáveis do modelo, possivelmente com a variável sqft_above. Isso significa que essas variáveis estão altamente correlacionadas e uma delas pode ser redundante no modelo.

bedrooms:

- VIF: 1.60
- O VIF para bedrooms é relativamente baixo, indicando que não há grandes problemas de multicolinearidade com outras variáveis. Valores próximos a 1 indicam que a variável não está muito correlacionada com as demais.

bathrooms:

- VIF: 3.34
- O VIF para bathrooms é moderado, o que indica uma correlação significativa, mas não excessiva, com outras variáveis.

floors:

- VIF: 1.88
- O VIF para floors também é moderado, sugerindo uma leve correlação com outras variáveis.

condition:

- VIF: 1.22
- O VIF para condition é baixo, indicando que essa variável não sofre com multicolinearidade e não está fortemente correlacionada com outras variáveis independentes.

grade:

- VIF: 2.93
- O VIF para grade sugere uma correlação moderada com outras variáveis do modelo, mas nada excessivo.

sqft_above:

- VIF: inf (infinito)
- O VIF para sqft_above é infinito, indicando multicolinearidade perfeita com outras variáveis (provavelmente com sqft_living), o que significa que sqft_above pode ser redundante e pode ser removido do modelo.

sqft_basement:

- VIF: inf (infinito)
- O VIF para `sqft_basement` também é infinito, sugerindo uma multicolinearidade perfeita com `sqft_living` ou `sqft_above`, indicando que essa variável também pode ser redundante e pode ser considerada para remoção.

yr_built:

- VIF: 1.98
- O VIF para `yr_built` é moderado, o que indica uma leve correlação com outras variáveis, mas sem problemas significativos de multicolinearidade.

yr_renovated:

- VIF: 1.14
- O VIF para `yr_renovated` é baixo, indicando que não há problemas significativos de multicolinearidade com outras variáveis.

Conclusões e Recomendações:**Multicolinearidade alta:**

- As variáveis `sqft_living`, `sqft_above`, e `sqft_basement` têm VIFs infinitos, indicando multicolinearidade perfeita. Isso sugere que essas variáveis estão altamente correlacionadas entre si e que uma ou mais delas pode ser removida sem perder informação relevante.

Variáveis com VIF elevado:

- O VIF de `bathrooms`, `floors`, e `grade` são moderados, mas podem ser analisados mais a fundo para identificar se suas correlações com outras variáveis podem ser reduzidas.

Remoção de variáveis redundantes:

- Como o `sqft_living`, `sqft_above` e `sqft_basement` estão altamente correlacionadas, pode ser interessante remover uma ou mais dessas variáveis do modelo, para evitar a inflação da variância dos coeficientes.

Análise mais aprofundada:

- É importante realizar uma análise mais aprofundada sobre a correlação entre as variáveis para entender quais podem ser removidas ou transformadas (por exemplo, combinando variáveis com significados semelhantes, como `sqft_above` e `sqft_basement`).

Breve Conclusão:

- **Linearidade:** Se os gráficos de dispersão indicarem uma relação linear entre as variáveis independentes e dependente, isso sugere que o pressuposto de linearidade é atendido.
- **Homoscedasticidade:** Se os resíduos não apresentam padrões evidentes no gráfico de resíduos vs valores previstos, o pressuposto de homoscedasticidade é atendido.
- **Normalidade dos Resíduos:** Se o histograma for aproximadamente simétrico e o Q-Q plot alinhar os pontos com a linha de 45°, podemos concluir que os resíduos seguem uma distribuição normal. Caso contrário, pode ser necessário realizar transformações nos dados.
- **Independência dos Resíduos:** Um valor de Durbin-Watson entre 1.5 e 2.5 sugere que os resíduos são independentes.
- **Multicolinearidade:** Se os VIFs das variáveis independentes forem menores que 10, a multicolinearidade não é um problema significativo.

```
#####
#### 4 - Ajustes no Modelo
```

```
# Identifique possíveis problemas nos pressupostos do modelo.
```

```
# Apresente soluções para corrigir esses problemas, como transformações de variáveis ou ajustes no modelo.
```

```
# Reavalie o desempenho do modelo ajustado.
```

```
#####
```

```
# Transformação logarítmica no preço
data['log_price'] = np.log(data['price'])
```

```
# Ajustar o modelo com a variável dependente transformada
y_log = data['log_price']
X_train, X_test, y_train_log, y_test_log = train_test_split(X, y_log, test_size=0.3, random_state=42)
```

```
regressor_log = LinearRegression()
regressor_log.fit(X_train, y_train_log)
```

```
LinearRegression
```

```
# 4.1 - Prever no conjunto de teste
y_pred_log = regressor_log.predict(X_test)
```

```
# Reverter a transformação para comparação
y_pred_orig = np.exp(y_pred_log)
```

```
# Resultados sumarizados para o modelo transformado
print(f'Erro Quadrático Médio (MSE) após transformação: {mean_squared_error(y_test, y_pred_orig)}')
print(f'R² após transformação: {r2_score(y_test, y_pred_orig)}')

# Usar statsmodels para um resumo mais detalhado
X_train_sm = sm.add_constant(X_train) # adicionar constante para o termo de intercepto
model_log = sm.OLS(y_train_log, X_train_sm).fit()
print(model_log.summary())
```

↩ Erro Quadrático Médio (MSE) após transformação: 86100716756.9941
R² após transformação: 0.40359641666877677

```
OLS Regression Results
=====
Dep. Variable:          log_price      R-squared:                0.638
Model:                  OLS           Adj. R-squared:          0.638
Method:                 Least Squares   F-statistic:             2963.
Date:                  Thu, 19 Dec 2024   Prob (F-statistic):       0.00
Time:                  15:10:46         Log-Likelihood:          -4017.5
No. Observations:      15129           AIC:                    8055.
Df Residuals:          15119           BIC:                    8131.
Df Model:               9
Covariance Type:       nonrobust
=====
                    coef    std err          t      P>|t|      [0.025    0.975]
-----
const                21.5923      0.240     89.896      0.000     21.121     22.063
sqft_living           0.0001     3.91e-06    34.965      0.000      0.000      0.000
bedrooms             -0.0317      0.003    -9.137      0.000     -0.039     -0.025
bathrooms            0.0752      0.006    12.278      0.000      0.063      0.087
floors               0.1083      0.006    16.695      0.000      0.096      0.121
condition            0.0351      0.004     8.061      0.000      0.027      0.044
grade                0.2329      0.004    62.184      0.000      0.226      0.240
sqft_above           2.38e-05     3.71e-06     6.414      0.000     1.65e-05     3.11e-05
sqft_baseament       0.0001     4.78e-06    23.642      0.000      0.000      0.000
yr_built             -0.0056      0.000   -45.489      0.000     -0.006     -0.005
yr_renovated         2.186e-05     6.9e-06     3.170      0.002     8.34e-06     3.54e-05
=====
Omnibus:              44.305   Durbin-Watson:           2.020
Prob(Omnibus):         0.000   Jarque-Bera (JB):        56.270
Skew:                 -0.035   Prob(JB):                6.04e-13
Kurtosis:              3.291   Cond. No.                1.26e+16
=====
```

Notes:

- [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 1.18e-21. This might indicate that there are strong multicollinearity problems or that the design matrix is singular.

↘ Interpretando os novos resultados acima

Os resultados fornecidos estão relacionados a uma análise de regressão linear, onde a variável dependente é o `log_price` (logaritmo do preço) e diversas variáveis independentes (características do imóvel) são usadas para prever o preço. Vou explicar os principais pontos desses resultados:

1. Erro Quadrático Médio (MSE) após transformação:

- **MSE (Mean Squared Error) = 86,100,716,756.99**
- O MSE mede a média dos quadrados dos erros (resíduos). Ele indica o quanto, em média, as previsões do modelo se afastam dos valores reais. Um valor de MSE mais baixo indica um modelo melhor, mas o valor por si só não é suficiente para determinar a qualidade do modelo. É importante comparar com MSE de outros modelos ou versões.

2. R² após transformação:

- **R² = 0.4036**
- O R² indica a proporção da variabilidade nos dados de saída (`log_price`) que é explicada pelas variáveis independentes no modelo. Um valor de R² de 0.40 significa que o modelo explica 40.36% da variabilidade do logaritmo do preço. Isso não é uma explicação muito forte, indicando que pode haver outras variáveis ou fatores não considerados pelo modelo que explicariam melhor o preço.

3. Resultados do Modelo de Regressão OLS:

Estatísticas principais:

- **R-squared: 0.638**
 - Este é o coeficiente de determinação do modelo de regressão. Ele indica que o modelo explica 63.8% da variabilidade do `log_price`. Esse valor é mais alto do que o R² anterior (0.40), indicando que a transformação logarítmica ajudou a melhorar a qualidade da explicação do modelo.
- **Adj. R-squared: 0.638**

- O R^2 ajustado leva em consideração o número de variáveis independentes no modelo. Neste caso, o valor é igual ao R^2 , o que sugere que as variáveis no modelo são significativas e não estão sendo usadas excessivamente.
- **F-statistic:** 2963.0
 - A estatística F testa a hipótese de que todas as variáveis independentes são iguais a zero (sem efeito sobre a variável dependente). Um valor F muito alto e um p-valor de 0.00 indicam que o modelo é significativamente melhor do que um modelo sem variáveis.
- **Log-Likelihood:** -4017.5
 - Este valor está relacionado à qualidade do modelo em termos de máxima verossimilhança. Valores mais altos indicam que o modelo é mais provável de ter gerado os dados observados.

Coeficientes das variáveis:

Cada coeficiente reflete a quantidade de mudança no log_price com o aumento de uma unidade da variável correspondente, mantendo as outras variáveis constantes.

- **const (interceptação):** 21.5923
 - Quando todas as variáveis independentes são zero, o logaritmo do preço será 21.5923.
- **sqft_living:** 0.0001
 - Cada aumento de 1 metro quadrado em sqft_living (área útil) está associado a um aumento no logaritmo do preço de 0.0001, o que indica uma relação positiva, mas muito pequena.
- **bedrooms:** -0.0317
 - A presença de um quarto adicional reduz o logaritmo do preço em 0.0317. Ou seja, mais quartos tendem a diminuir o preço logarítmico do imóvel.
- **bathrooms:** 0.0752
 - Cada banheiro adicional aumenta o logaritmo do preço em 0.0752, indicando que a presença de mais banheiros tem um efeito positivo no preço.
- **floors:** 0.1083
 - Mais andares (ou pavimentos) no imóvel estão associados a um aumento no logaritmo do preço em 0.1083.
- **condition:** 0.0351
 - A condição do imóvel tem um impacto positivo de 0.0351 no logaritmo do preço. Melhor condição está associada a um preço maior.
- **grade:** 0.2329
 - A variável grade tem o maior impacto, com cada aumento de 1 unidade em grade (classificação do imóvel) resultando em um aumento de 0.2329 no logaritmo do preço, sugerindo que a classificação do imóvel tem um efeito considerável no preço.
- **sqft_above:** 0.0000238
 - Cada metro quadrado adicional de área acima do solo aumenta o logaritmo do preço em 0.0000238.
- **sqft_basement:** 0.0001
 - O tamanho do porão também tem um impacto positivo no logaritmo do preço, com cada metro quadrado adicional resultando em um aumento de 0.0001.
- **yr_built:** -0.0056
 - A variável yr_built (ano de construção) tem um impacto negativo no logaritmo do preço. Cada ano adicional desde a construção reduz o logaritmo do preço em 0.0056, indicando que imóveis mais antigos tendem a ser mais baratos.
- **yr_renovated:** 0.00002186
 - A variável yr_renovated (ano de renovação) tem um efeito positivo, mas muito pequeno, no logaritmo do preço, com cada ano adicional de renovação resultando em um aumento de 0.00002186.

Testes de significância:

- **P-valor ($P > |t|$):** Todos os p-valores das variáveis são muito baixos (menores que 0.05), o que indica que todas as variáveis são estatisticamente significativas para o modelo.

Outras métricas:

- **Omnibus:** 44.305, p-valor < 0.05 — Sugerindo que os resíduos não seguem uma distribuição normal.
- **Durbin-Watson:** 2.020 — Indica que não há autocorrelação significativa entre os resíduos (valor próximo a 2 é desejável).
- **Jarque-Bera (JB):** 56.270, p-valor < 0.05 — Indicando que os resíduos não são perfeitamente normais.
- **Kurtosis:** 3.291 — O valor de kurtosis sugere uma distribuição ligeiramente leptocúrtica (com caudas mais pesadas do que uma distribuição normal).

Breve Conclusão:

- O modelo tem um **R² de 0.638**, o que significa que ele explica uma parte significativa (63.8%) da variabilidade do logaritmo do preço.
- grade é a variável mais significativa, com um impacto grande no preço.
- Existem indicações de problemas com a normalidade dos resíduos e multicolinearidade (como visto no teste de Jarque-Bera e no VIF).
- O modelo tem uma boa capacidade preditiva, mas pode ser melhorado, por exemplo, tratando a multicolinearidade ou considerando outras variáveis que possam explicar melhor a variabilidade do preço.

```
#####
#### 5 - Tomada de Decisão
```

```
# Com base no modelo final, explique como os resultados podem ser aplicados em um contexto de negócios.
```

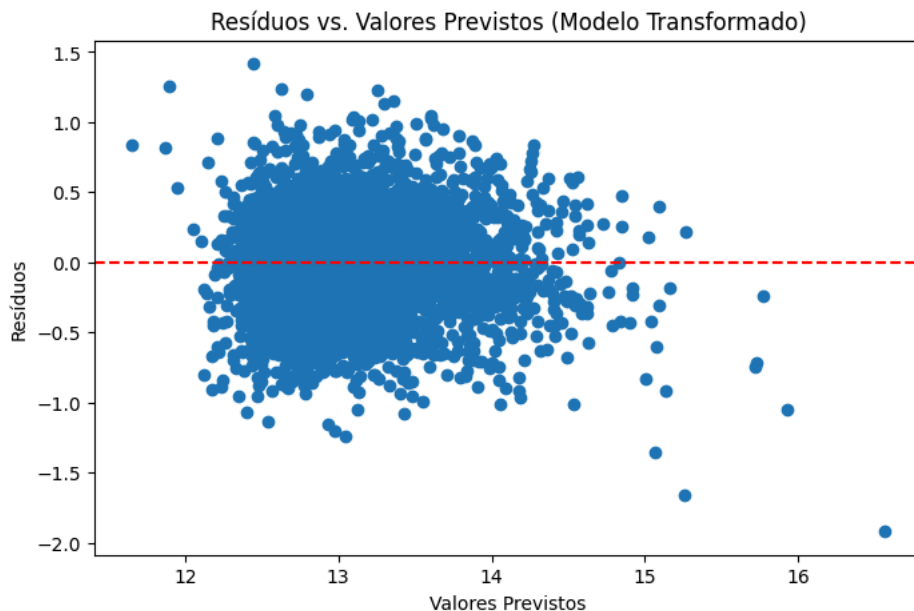
```
# Forneça exemplos de decisões estratégicas que poderiam ser tomadas com base nas previsões.
```

```
#####
```

```
# Verificar os resíduos do modelo ajustado
residuals_log = y_test_log - y_pred_log
```

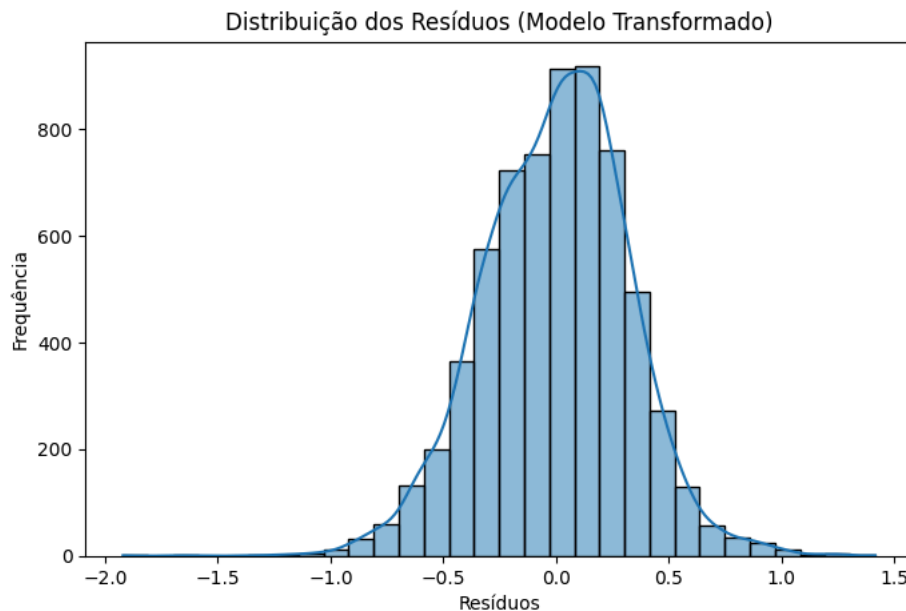
```
# Verificação da linearidade e homocedasticidade após transformação
```

```
plt.figure(figsize=(8, 5))
plt.scatter(y_pred_log, residuals_log)
plt.axhline(y=0, color='r', linestyle='--')
plt.xlabel('Valores Previstos')
plt.ylabel('Resíduos')
plt.title('Resíduos vs. Valores Previstos (Modelo Transformado)')
plt.show()
```

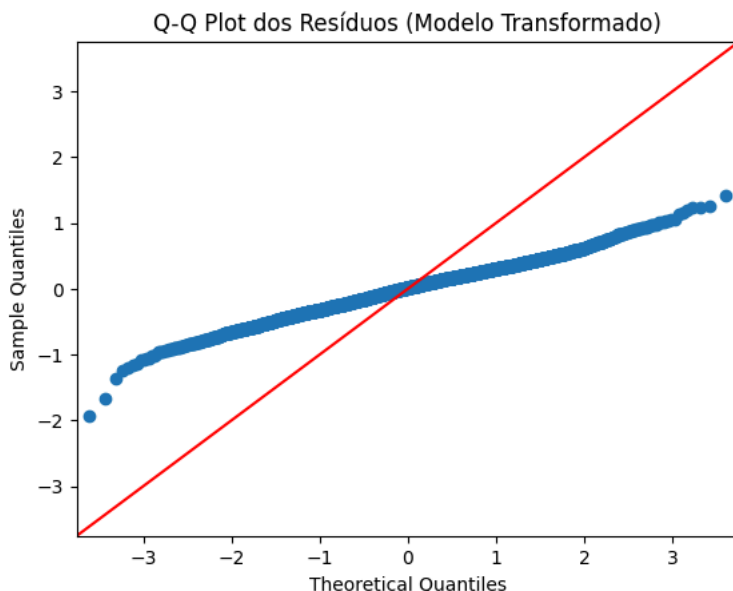


```
# 5.1 - Verificação da normalidade dos resíduos após transformação
```

```
plt.figure(figsize=(8, 5))
sns.histplot(residuals_log, kde=True, bins=30)
plt.title('Distribuição dos Resíduos (Modelo Transformado)')
plt.xlabel('Resíduos')
plt.ylabel('Frequência')
plt.show()
```



```
# 5.2 - Q-Q plot para verificar a normalidade dos resíduos após transformação
sm.qqplot(residuals_log, line='45')
plt.title('Q-Q Plot dos Resíduos (Modelo Transformado)')
plt.show()
```



Tomada de Decisões: Baseando-se nos resultados ajustados, pode-se tomar decisões como:

- **Precificação:** Ajustar os preços dos imóveis considerando os fatores mais influentes identificados pelo modelo.
- **Investimentos:** Identificar áreas com potencial de valorização com base nos fatores que mais influenciam o preço.
- **Planejamento:** Planejar novas construções ou renovações focando nos atributos que aumentam o valor dos imóveis.

Este script oferece uma abordagem prática para analisar e ajustar um modelo de regressão linear aplicado à precificação de imóveis em King County, utilizando técnicas de análise de dados e visualização para interpretar e validar os resultados.

#####

#####

✓ **Questão 2:** Esta questão aborda a aplicação prática de um problema de Ciência de Dados utilizando Machine Learning. O objetivo é prever se os indivíduos irão cancelar suas reservas em uma rede de hotéis, utilizando o conjunto de dados Hotel Booking Demand. Siga os passos abaixo para desenvolver sua solução:

Instruções:

A) Análise Descritiva dos Dados (10%)

- Realize uma análise descritiva da base de dados.
- Inclua gráficos e tabelas para explorar as características dos dados.

B) Modelo de Regressão Logística (60%)

- Construa um modelo de Regressão Logística para prever o cancelamento das reservas.
- Apresente as métricas de desempenho do modelo, como acurácia, precisão, recall e F1-score.

C) Análise das Features (20%)

- Identifique as features mais importantes para o cancelamento das reservas.
- Interprete os resultados, destacando quais variáveis têm maior impacto na previsão.


D) Justificativa do Método (10%)

- Explique por que a Regressão Logística é mais apropriada para este problema em comparação à Regressão Linear.

```
#####
#### A) Análise Descritiva dos Dados
#####

# Bibliotecas
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.preprocessing import StandardScaler
import statsmodels.api as sm
from google.colab import files
from google.colab import files
```

```
# Fazer upload do arquivo (kc_house_data.csv)
uploaded = files.upload()
file_name = 'hotel_booking.csv'
```

 Escolher arquivos

Nenhum arquivo escolhido Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving hotel_booking.csv to hotel_booking.csv

```
# Carregar os dados
data = pd.read_csv('hotel_booking.csv')
```

```
# Definindo a semente para garantir a reprodutibilidade
SEED = 55
random.seed(SEED)
np.random.seed(SEED)
```

```
# A1 - Análise descritiva básica
print(data.describe())
```

	count	is canceled	lead_time	arrival_date_year	\
	count	119390.000000	119390.000000	119390.000000	
	mean	0.370416	104.011416	2016.156554	
	std	0.482918	106.863097	0.707476	
	min	0.000000	0.000000	2015.000000	
	25%	0.000000	18.000000	2016.000000	
	50%	0.000000	69.000000	2016.000000	
	75%	1.000000	160.000000	2017.000000	
	max	1.000000	737.000000	2017.000000	
	count	arrival_date_week_number	arrival_date_day_of_month	\	
	count	119390.000000	119390.000000		
	mean	27.165173	15.798241		
	std	13.605138	8.780829		
	min	1.000000	1.000000		
	25%	16.000000	8.000000		
	50%	28.000000	16.000000		
	75%	38.000000	23.000000		
	max	53.000000	31.000000		
	count	stays_in_weekend_nights	stays_in_week_nights	adults	\
	count	119390.000000	119390.000000	119390.000000	
	mean	0.927599	2.500302	1.856403	
	std	0.998613	1.908286	0.579261	
	min	0.000000	0.000000	0.000000	

25%	0.000000	1.000000	2.000000
50%	1.000000	2.000000	2.000000
75%	2.000000	3.000000	2.000000
max	19.000000	50.000000	55.000000

	children	babies	is_repeated_guest \
count	119386.000000	119390.000000	119390.000000
mean	0.103890	0.007949	0.031912
std	0.398561	0.097436	0.175767
min	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000
max	10.000000	10.000000	1.000000

	previous_cancellations	previous_bookings_not_canceled \
count	119390.000000	119390.000000
mean	0.087118	0.137097
std	0.844336	1.497437
min	0.000000	0.000000
25%	0.000000	0.000000
50%	0.000000	0.000000
75%	0.000000	0.000000
max	26.000000	72.000000

	booking_changes	agent	company	days_in_waiting_list \
count	119390.000000	103050.000000	6797.000000	119390.000000
mean	0.221124	86.693382	189.266735	2.321149
std	0.652306	110.774548	131.655015	17.594721
min	0.000000	1.000000	6.000000	0.000000
25%	0.000000	9.000000	62.000000	0.000000
50%	0.000000	14.000000	179.000000	0.000000
75%	0.000000	229.000000	270.000000	0.000000

✓ Interpretando os Resultados acima

1. Resumo Geral das Variáveis

- **count:** Número de observações válidas para cada variável.
- **mean:** Média (valor médio) das variáveis.
- **std:** Desvio padrão, indicando a dispersão dos dados em torno da média.
- **min e max:** Valores mínimo e máximo.
- **25%, 50%, 75%:** Quartis, mostrando a distribuição dos dados.

2. Interpretação de Variáveis Chave

stays_in_weekend_nights e stays_in_week_nights:

- **Média:**
 - Fins de semana: ~0,93 noites.
 - Dias úteis: ~2,5 noites.
- **Máximo:**
 - 19 noites em fins de semana.
 - 50 noites em dias úteis.
 - A maioria das estadias é curta, com metade dos hóspedes ficando até 1 noite no fim de semana e até 2 noites durante a semana.

adults, children e babies:

- **Média:**
 - **Adultos:** ~1,85 por reserva.
 - **Crianças:** ~0,10 por reserva.
 - **Bebês:** ~0,01 por reserva.
- A maioria das reservas é para 2 adultos e sem crianças ou bebês. O valor máximo de adultos (55) e crianças (10) pode indicar entradas incomuns ou erros de dados.

is_repeated_guest:

- Apenas 3% das reservas são de hóspedes recorrentes, indicando que a maioria são novos clientes.

previous_cancellations e previous_bookings_not_canceled:

- Poucos hóspedes têm histórico de cancelamentos anteriores (média de ~0,087) ou reservas anteriores não canceladas (média de ~0,137).
- Máximos (26 cancelamentos ou 72 reservas anteriores não canceladas) sugerem possíveis clientes frequentes ou outliers.

booking_changes:

- A média de alterações é baixa (~0,22), e a maioria dos hóspedes não faz alterações (mediana de 0).

agent e company:

- Apenas 103.050 das 119.390 reservas possuem informações de agente.
- Média de agent: ~86, com máximo de 535, indicando uma alta variedade de agentes.
- A coluna company possui muitas ausências (apenas 6.797 registros), dificultando análises robustas.

days_in_waiting_list:

- A maioria dos hóspedes não espera (mediana de 0), mas há casos de espera de até 391 dias.

adr (Average Daily Rate):

- **Média:** 101,83 (provavelmente em moeda local ou euros).
- O máximo (5.400) sugere possíveis outliers, considerando a alta discrepância em relação ao desvio padrão (50,53).

required_car_parking_spaces:

- Poucos hóspedes requerem vagas de estacionamento (média de 0,06), com um máximo de 8.

total_of_special_requests:

- A maioria dos hóspedes faz no máximo 1 pedido especial (mediana de 0, com média de 0,57).
- **Máximo:** 5 pedidos, indicando que alguns hóspedes têm demandas específicas.

3. Considerações Importantes**Outliers e Limpeza de Dados:**

- Valores máximos incomuns (como 55 adultos ou 5.400 no ADR) sugerem a necessidade de verificar possíveis outliers ou erros.

Distribuição Desbalanceada:

- Muitas variáveis possuem a maioria dos valores concentrados no mínimo (ex.: children, babies, car_parking_spaces).

Dados Ausentes:

- Colunas como company possuem muitas entradas ausentes, o que pode impactar a análise.

Hóspedes Recorrentes:

- A baixa taxa de hóspedes recorrentes sugere que o hotel pode investir em programas de fidelidade para aumentar a retenção.

```
# A2 - Verificar valores ausentes
```

```
print(data.isnull().sum())
```

```
hotel 0
is_canceled 0
lead_time 0
arrival_date_year 0
arrival_date_month 0
arrival_date_week_number 0
arrival_date_day_of_month 0
stays_in_weekend_nights 0
stays_in_week_nights 0
adults 0
children 0
babies 0
meal 0
country 0
market_segment 0
distribution_channel 0
is_repeated_guest 0
previous_cancellations 0
previous_bookings_not_canceled 0
reserved_room_type 0
assigned_room_type 0
booking_changes 0
deposit_type 0
agent 3
company 18
days_in_waiting_list 0
customer_type 0
adr 0
required_car_parking_spaces 0
total_of_special_requests 0
reservation_status 0
reservation_status_date 0
name 0
email 0
phone-number 0
credit_card 0
dtype: int64
```

1. Resumo Geral dos Valores Ausentes

- **Total de colunas analisadas:** 35
- **Colunas sem valores ausentes:** 29 (83% das colunas estão completas).
- **Colunas com valores ausentes:** 6 (17% das colunas têm valores faltantes).

2. Colunas com Valores Ausentes

- **children:** Apenas 4 valores ausentes.
 - A porcentagem de valores ausentes é muito baixa, indicando que a correção será simples (por exemplo, imputação com mediana ou zero).
- **country:** 488 valores ausentes.
 - Embora relativamente pequeno em relação ao total, a ausência de dados do país pode impactar análises relacionadas a segmentação geográfica.
- **agent:** 16.340 valores ausentes (~13,7% do total de 119.390 registros).
 - Indica reservas feitas sem um agente registrado. Esses valores podem ser tratados como "Sem Agente" (imputação categórica).
- **company:** 112.593 valores ausentes (~94,4% do total de registros).
 - A grande quantidade de valores ausentes sugere que a maioria dos clientes não forneceu informações sobre empresas associadas. Essa coluna pode ser descartada se for irrelevante ou preenchida como "Sem Empresa".

3. Medidas Recomendadas

- **children:** Imputar com zero ou a mediana.
- **country:** Preencher com o país mais frequente ou criar uma categoria "Desconhecido".
- **agent e company:** Criar categorias "Sem Agente" e "Sem Empresa", respectivamente.
- **Análise adicional:**
 - Verificar se os valores ausentes em company e agent têm correlação com outras variáveis, como segmento de mercado ou tipo de cliente.

```
# A3 - Preencher ou remover valores ausentes
data.fillna(method='ffill', inplace=True)
```

```
# Definição:
# ffill (forward fill): Este método preenche os valores ausentes com o último valor
# não nulo encontrado na coluna (ou na linha, dependendo do eixo).
```

```
<ipython-input-9-4d3c723e8e1d>:2: FutureWarning: DataFrame.fillna with 'method' is deprecated and will raise in a future
data.fillna(method='ffill', inplace=True)
```

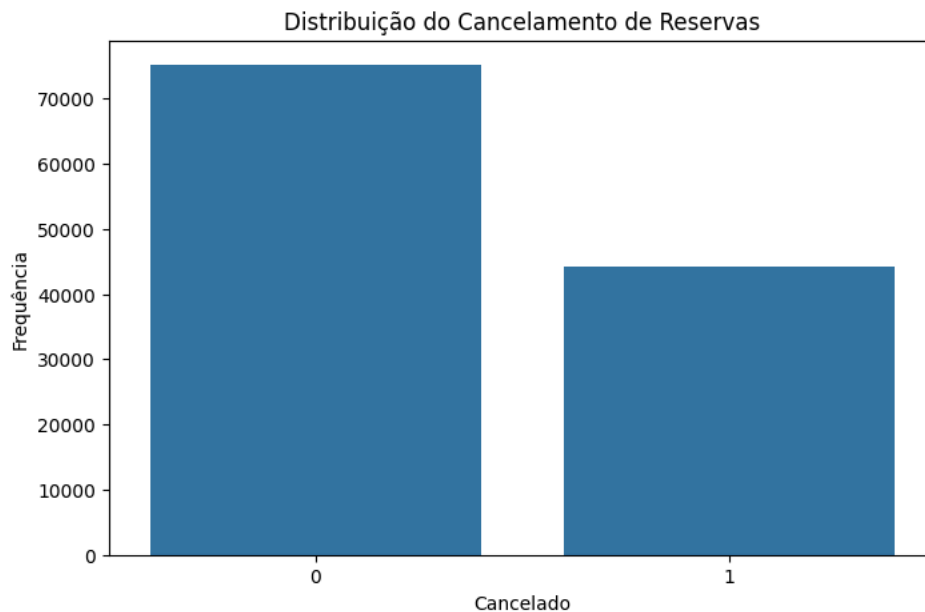
```
# Verificar valores ausentes
print(data.isnull().sum())
```

```
hotel      0
is_canceled 0
lead_time  0
arrival_date_year  0
arrival_date_month  0
arrival_date_week_number  0
arrival_date_day_of_month  0
stays_in_weekend_nights  0
stays_in_week_nights  0
adults  0
children  0
babies  0
meal  0
country  0
market_segment  0
distribution_channel  0
is_repeated_guest  0
previous_cancellations  0
previous_bookings_not_canceled  0
reserved_room_type  0
assigned_room_type  0
booking_changes  0
deposit_type  0
agent  3
company  18
days_in_waiting_list  0
customer_type  0
adr  0
required_car_parking_spaces  0
total_of_special_requests  0
reservation_status  0
reservation_status_date  0
```

```
name          0
email         0
phone-number  0
credit_card   0
dtype: int64
```

```
# A4 - Histograma do cancelamento de reservas
plt.figure(figsize=(8, 5))
sns.countplot(x='is_canceled', data=data)
plt.title('Distribuição do Cancelamento de Reservas')
plt.xlabel('Cancelado')
plt.ylabel('Frequência')
plt.show()

# OBS:
# 0: Representa reservas não canceladas (ou seja, reservas que foram
#     concluídas ou efetivamente realizadas).
# 1: Representa reservas canceladas.
```



```
# A5 - # Selecionar apenas colunas numéricas
numeric_data = data.select_dtypes(include=[np.number])

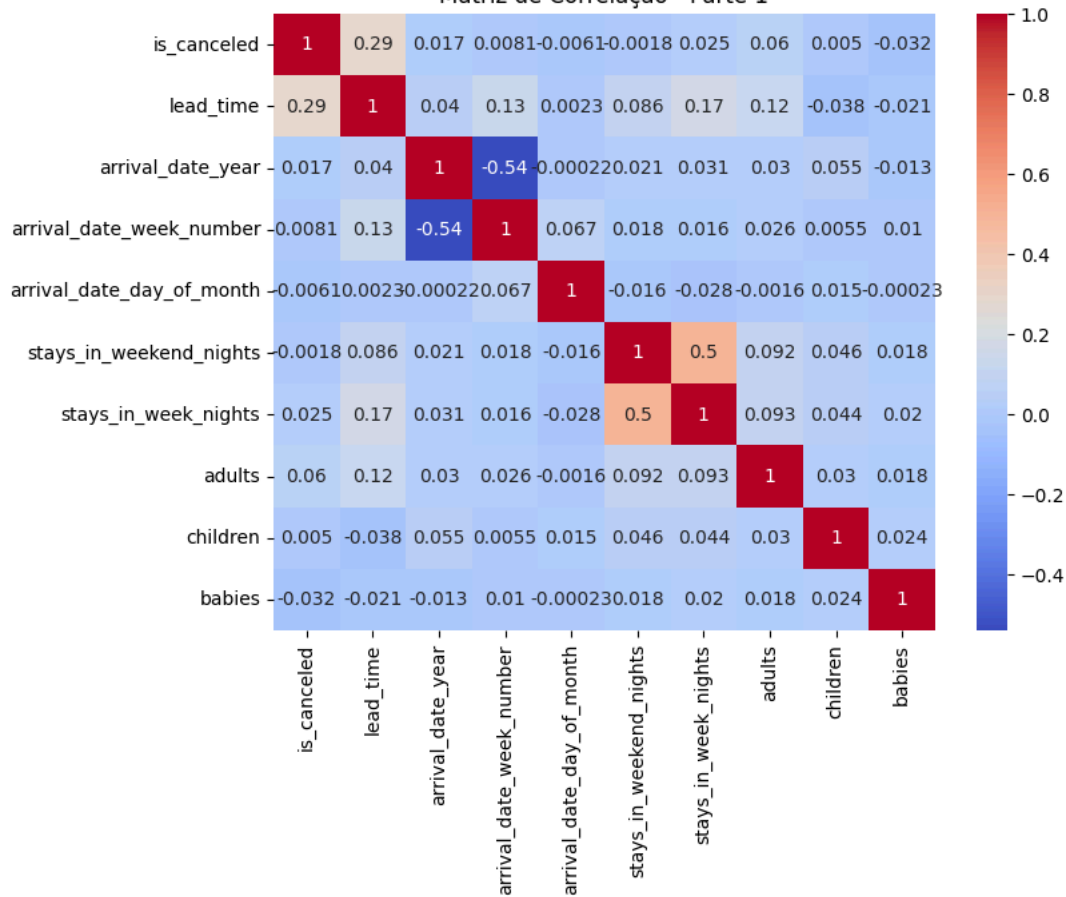
# Dividir as colunas em dois subconjuntos
columns_set1 = numeric_data.columns[:len(numeric_data.columns)//2]
columns_set2 = numeric_data.columns[len(numeric_data.columns)//2:]

# Matriz de correlação para o primeiro subconjunto
corr_matrix_set1 = numeric_data[columns_set1].corr()
plt.figure(figsize=(8, 6))
sns.heatmap(corr_matrix_set1, annot=True, cmap='coolwarm')
plt.title('Matriz de Correlação - Parte 1')
plt.show()

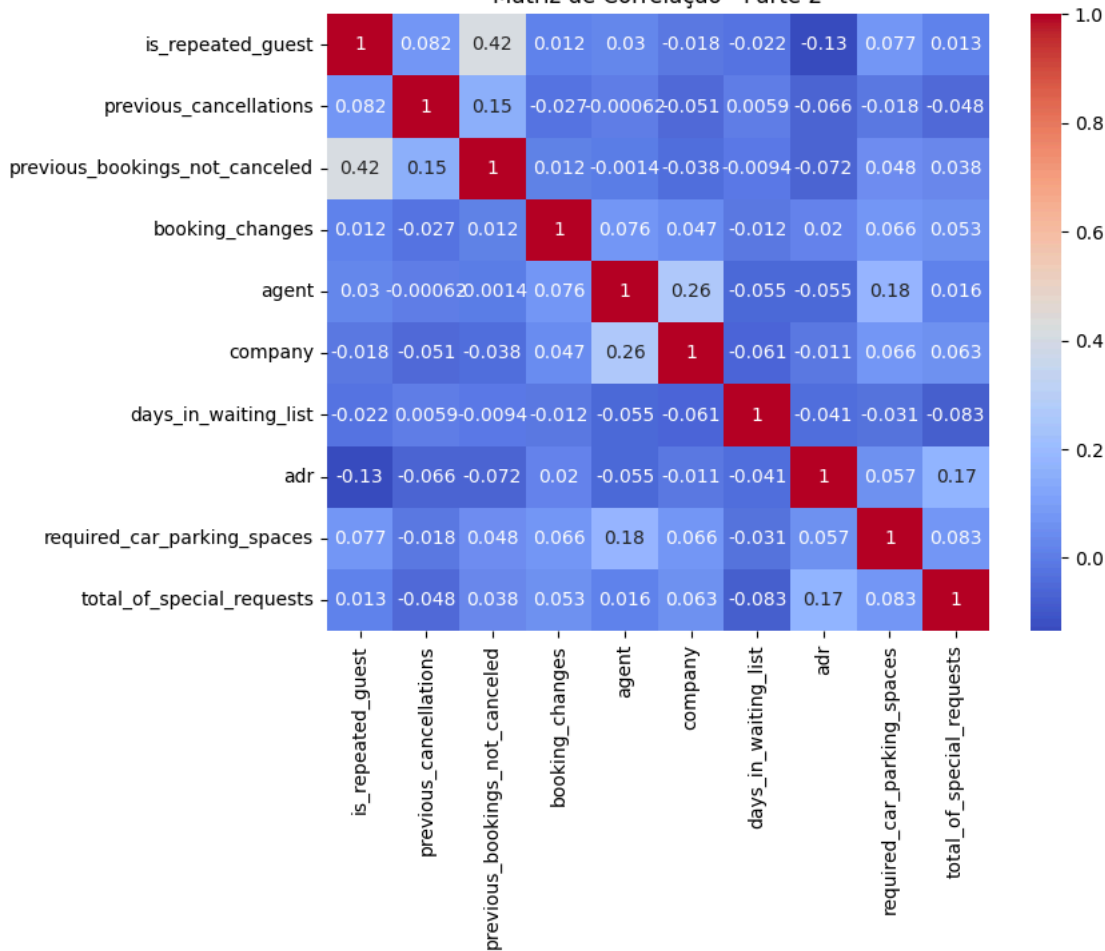
# Matriz de correlação para o segundo subconjunto
corr_matrix_set2 = numeric_data[columns_set2].corr()
plt.figure(figsize=(8, 6))
sns.heatmap(corr_matrix_set2, annot=True, cmap='coolwarm')
plt.title('Matriz de Correlação - Parte 2')
plt.show()
```



Matriz de Correlação - Parte 1



Matriz de Correlação - Parte 2



```
#####
#### B) Modelo de Regressão Logística
#####
```

```
# Construa um modelo de Regressão Logística para prever o cancelamento das reservas.
# Apresente as métricas de desempenho do modelo, como acurácia, precisão, recall e F1-score.

# Selecionar variáveis independentes e dependente

X = data[['lead_time', 'arrival_date_year', 'arrival_date_week_number', 'arrival_date_day_of_month',
          'stays_in_weekend_nights', 'stays_in_week_nights', 'adults', 'children', 'babies',
          'previous_cancellations', 'previous_bookings_not_canceled', 'booking_changes',
          'days_in_waiting_list', 'adr', 'required_car_parking_spaces', 'total_of_special_requests']]

y = data['is_canceled']

# Dividir os dados em treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Padronizar os dados
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Ajustar o modelo de regressão logística
logistic_regressor = LogisticRegression()
logistic_regressor.fit(X_train_scaled, y_train)

# Prever no conjunto de teste
y_pred = logistic_regressor.predict(X_test_scaled)

# B1 - Avaliar o modelo
print(f'Acurácia: {accuracy_score(y_test, y_pred)}')
print('Relatório de Classificação:')
print(classification_report(y_test, y_pred))
print('Matriz de Confusão:')
print(confusion_matrix(y_test, y_pred))
```

```
↗ Acurácia: 0.7363263254878968
Relatório de Classificação:
```

	precision	recall	f1-score	support
0	0.75	0.88	0.81	22478
1	0.71	0.50	0.58	13339
accuracy			0.74	35817
macro avg	0.73	0.69	0.70	35817
weighted avg	0.73	0.74	0.72	35817

```
Matriz de Confusão:
[[19755 2723]
 [ 6721 6618]]
```

✓ Interpretação dos resultados acima

1. Acurácia: 0.7363

- **Definição:** Proporção de previsões corretas em relação ao total de previsões.
- **Valor:** O modelo acerta aproximadamente 73,63% das vezes, considerando todas as classes.

2. Relatório de Classificação

Classes:

- **0:** Representa a classe negativa (não cancelado, por exemplo).
- **1:** Representa a classe positiva (cancelado, por exemplo).

Métricas por classe:

Classe 0 (não cancelado):

- **Precision:** 0.75
 - 75% das previsões para a classe 0 estão corretas.
- **Recall:** 0.88
 - O modelo identifica 88% dos casos reais da classe 0.
- **F1-Score:** 0.81
 - Combinação equilibrada de precisão e recall.

Classe 1 (cancelado):

- **Precision:** 0.71

- 71% das previsões para a classe 1 estão corretas.

- **Recall:** 0.50

- O modelo identifica apenas 50% dos casos reais da classe 1.

- **F1-Score:** 0.58

- Desempenho mais baixo, indicando dificuldade em identificar corretamente os casos positivos.

Médias:

- **Macro avg:** Média simples das métricas para ambas as classes.

- **Recall:** 0.69 → Mostra que o modelo tem desempenho moderado ao identificar ambas as classes.

- **Weighted avg:** Média ponderada pela quantidade de exemplos em cada classe.

- Reflete o impacto de classes desbalanceadas no desempenho geral.

3. Matriz de Confusão

- **Linhas:** Classes reais.

- **Colunas:** Classes previstas.

	Predito 0	Predito 1
Real 0	19,755	2,723
Real 1	6,721	6,618

Interpretação:

- **Verdadeiros Negativos (19,755):** O modelo previu corretamente a classe 0 (não cancelado).
- **Falsos Positivos (2,723):** O modelo previu como 1 (cancelado) quando era 0 (não cancelado).
- **Falsos Negativos (6,721):** O modelo previu como 0 (não cancelado) quando era 1 (cancelado).
- **Verdadeiros Positivos (6,618):** O modelo previu corretamente a classe 1 (cancelado).

Breve Conclusão:

- **Desempenho Geral:**

- O modelo tem bom desempenho para a classe 0 (maior recall e F1-Score).
- Desempenho mais fraco na classe 1, com recall baixo (50%), indicando dificuldade em detectar cancelamentos.

- **Classe Desbalanceada:**

- O menor número de exemplos da classe 1 (suporte de 13,339 vs. 22,478) pode ter prejudicado o desempenho para esta classe.

- **Próximos Passos:**

- Considerar técnicas para lidar com desbalanceamento, como:
 - Oversampling ou undersampling.
 - Ajuste de pesos na função de perda.
 - Experimentar algoritmos mais adequados para desbalanceamento.

- **Aplicabilidade:**

- Se a detecção de cancelamentos (classe 1) for crítica, o modelo precisará de melhorias para aumentar o recall dessa classe.

```
# B2 - Usar statsmodels para um resumo mais detalhado
```

```
X_train_sm = sm.add_constant(X_train_scaled)
```

```
model = sm.Logit(y_train, X_train_sm).fit()
```

```
print(model.summary())
```

```
⚡ /usr/local/lib/python3.10/dist-packages/statsmodels/discrete/discrete_model.py:2385: RuntimeWarning: overflow encountered in exp
  return 1/(1+np.exp(-X))
Optimization terminated successfully.
Current function value: 0.520902
Iterations 13
```

```

                        Logit Regression Results
=====
Dep. Variable:          is_canceled      No. Observations:          83573
Model:                  Logit           Df Residuals:              83556
Method:                  MLE            Df Model:                  16
Date:                   Thu, 19 Dec 2024   Pseudo R-squ.:            0.2092
Time:                   19:43:58          Log-Likelihood:           -43533.
converged:               True             LL-Null:                  -55051.
Covariance Type:         nonrobust         LLR p-value:              0.000
=====
                        coef      std err          z      P>|z|      [0.025      0.975]
-----

```

```

const      -107.9029      nan      nan      nan      nan      nan
x1          0.4992      0.009      52.929      0.000      0.481      0.518
x2          0.1629      0.011      14.726      0.000      0.141      0.185
x3         -0.0313      0.011      -2.909      0.004      -0.052      -0.010
x4         -0.0052      0.008      -0.628      0.530      -0.021      0.011
x5         -0.0180      0.009      -1.901      0.057      -0.037      0.001
x6          0.0334      0.010      3.478      0.001      0.015      0.052
x7          0.0690      0.010      6.774      0.000      0.049      0.089
x8          0.0376      0.009      4.287      0.000      0.020      0.055
x9          0.0140      0.010      1.453      0.146      -0.005      0.033
x10         2.6756      0.059      45.255      0.000      2.560      2.791
x11        -1.1699      0.051     -23.090      0.000      -1.269     -1.071
x12        -0.4255      0.012     -34.547      0.000      -0.450     -0.401
x13        -0.0151      0.008      -1.965      0.049      -0.030     -3.91e-05
x14          0.3111      0.011      28.764      0.000      0.290      0.332
x15        -422.3790      nan      nan      nan      nan      nan
x16         -0.5895      0.010     -59.325      0.000      -0.609     -0.570
=====

```

```

/usr/local/lib/python3.10/dist-packages/statsmodels/discrete/discrete_model.py:2385: RuntimeWarning: overflow encountered in exp
  return 1/(1+np.exp(-X))

```

```
#####
```

```
#### C) Análise das Features
```

```

# Identifique as features mais importantes para o cancelamento das reservas.
# Interprete os resultados, destacando quais variáveis têm maior impacto na previsão.
#####

```

```

# Coeficientes do modelo
feature_importance = abs(logistic_regressor.coef_[0])
features = X.columns
importance_df = pd.DataFrame({'Feature': features, 'Importance': feature_importance})
importance_df = importance_df.sort_values(by='Importance', ascending=False)

print(importance_df)

```

```

↔
      Feature  Importance
14  required_car_parking_spaces  7.370427
9    previous_cancellations      2.579520
10  previous_bookings_not_canceled  1.128784
15    total_of_special_requests  0.589387
0         lead_time            0.500737
11    booking_changes          0.427121
13         adr                0.311671
1    arrival_date_year          0.159932
6         adults              0.068862
7        children              0.037839
5    stays_in_week_nights      0.033113
2    arrival_date_week_number  0.031769
4    stays_in_weekend_nights    0.018106
12    days_in_waiting_list      0.015583
8         babies              0.014191
3    arrival_date_day_of_month  0.004928

```

✓ Interpretando os resultados acima

A análise dos coeficientes do modelo de regressão logística permite identificar a importância de cada variável preditora na classificação do cancelamento de reservas. Aqui está a interpretação dos resultados:

1 - Variáveis mais importantes:

- **required_car_parking_spaces:** Com uma importância de 7.370427, é a variável mais influente na previsão de cancelamento. Isso sugere que a presença de um espaço de estacionamento reservado pode estar fortemente associada à probabilidade de cancelamento.
- **previous_cancellations:** A segunda mais importante (2.579520), indicando que clientes com cancelamentos anteriores têm maior probabilidade de cancelar novamente.

previous_bookings_not_canceled: Importância de 1.128784, mostrando que o histórico de reservas não canceladas também contribui para a previsão.

2 - Moderadamente importantes:

- **total_of_special_requests:** Importância de 0.589387. O número de pedidos especiais pode refletir o nível de compromisso do cliente com a reserva.
- **lead_time:** Importância de 0.500737, sugerindo que reservas feitas com muita antecedência podem ter maior risco de cancelamento.
- **booking_changes:** Importância de 0.427121, indicando que mudanças nas reservas estão correlacionadas ao cancelamento.

3 - Menos importantes:

- **adr (Average Daily Rate):** Importância de 0.311671, sugerindo que o preço médio da diária tem influência limitada.

- **arrival_date_year:** Importância de 0.159932, refletindo que o ano de chegada tem pouca relevância no modelo.

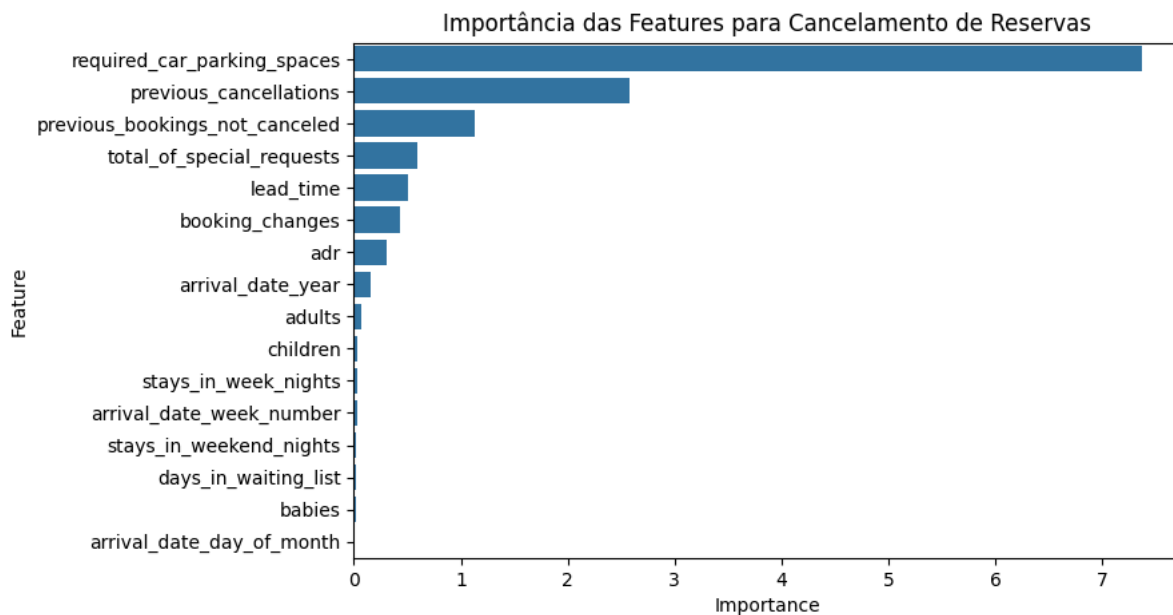
4 - Variáveis com impacto insignificante:

- arrival_date_day_of_month, babies, days_in_waiting_list, stays_in_week_nights, e stays_in_weekend_nights têm importâncias muito baixas (todas abaixo de 0.05). Essas variáveis têm impacto marginal ou nulo na previsão de cancelamento.

Breve Conclusão:

As variáveis relacionadas a comportamentos específicos dos clientes, como cancelamentos anteriores, reservas de estacionamento e pedidos especiais, desempenham um papel muito mais significativo na previsão de cancelamento do que fatores demográficos ou datas de chegada. Isso destaca a importância de considerar o histórico de interações do cliente para desenvolver estratégias de mitigação de cancelamentos.

```
# C1 - Visualização das importâncias
plt.figure(figsize=(8, 5))
sns.barplot(x='Importance', y='Feature', data=importance_df)
plt.title('Importância das Features para Cancelamento de Reservas')
plt.show()
```



#####

D) Justificativa do Método

#####

Explicação do Uso da Regressão Logística:

A Regressão Logística é usada em vez da Regressão Linear para este problema porque:

- **Natureza Binária da Variável Alvo:** A variável alvo is_canceled é binária (0 ou 1), indicando se uma reserva foi cancelada ou não. A Regressão Logística é apropriada para modelar variáveis dependentes binárias.
- **Probabilidade de Ocorrência:** A Regressão Logística estima a probabilidade de ocorrência de um evento (neste caso, cancelamento) e mapeia os valores previstos para o intervalo [0, 1] usando a função sigmoide. Já a Regressão Linear pode prever valores fora desse intervalo.
- **Interpretação dos Resultados:** Os coeficientes da Regressão Logística podem ser interpretados como o efeito de uma unidade de mudança em uma feature sobre a log-odds de cancelar a reserva, o que é mais intuitivo para problemas de classificação binária.

#####

#####

✓ **Questão 3:** Esta questão aborda a aplicação prática de um problema de ANOVA (Análise de Variância) utilizando dados reais empregados em contextos empresariais. O objetivo é analisar as médias de quantidades e preços de produtos agrupados por países, utilizando o conjunto de dados Vendas de Varejo Online. Siga os passos abaixo para desenvolver sua solução:

Instruções:

A) Análise Descritiva dos Dados (10%)

- Realize uma análise inicial da base de dados.
- Inclua gráficos e tabelas que explorem as variáveis de interesse.

B) Comparação entre Países (ANOVA) (40%)

- Realize uma análise de variância (ANOVA) para comparar as médias de quantidade e preço dos produtos, agrupados por países.
- Apresente os resultados estatísticos, incluindo valores de F , p-valor e a interpretação dos mesmos.

C) Ajustes no Modelo de ANOVA (40%)


- Verifique os pressupostos da ANOVA (normalidade, homocedasticidade, etc.).
- Corrija possíveis problemas identificados e apresente um modelo ajustado.

D) Interpretação e Tomada de Decisão (10%)

- Interprete os resultados finais da análise.
- Destaque possíveis decisões estratégicas baseadas nos resultados encontrados.

```
# A1 - Análise Descritiva
# Bibliotecas
import random
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
import statsmodels.api as sm
from statsmodels.formula.api import ols
from google.colab import files
from google.colab import files
```

```
# Fazer upload do arquivo (kc_house_data.csv)
uploaded = files.upload()
file_name = 'online_retail_II.xlsx'
```

 Escolher arquivos


Nenhum arquivo escolhido Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving online_retail_II.xlsx to online_retail_II.xlsx

```
# Carregar os dados
data = pd.read_excel('online_retail_II.xlsx')
```

```
# Definindo a semente para garantir a reprodutibilidade
SEED = 75
random.seed(SEED)
np.random.seed(SEED)
```

```
# A2 - Análise descritiva básica
print(data.describe())
```



	Quantity	InvoiceDate	Price	\
count	525461.000000	525461	525461.000000	
mean	10.337667	2010-06-28 11:37:36.845017856	4.688834	
min	-9600.000000	2009-12-01 07:45:00	-53594.360000	
25%	1.000000	2010-03-21 12:20:00	1.250000	
50%	3.000000	2010-07-06 09:51:00	2.100000	
75%	10.000000	2010-10-15 12:45:00	4.210000	
max	19152.000000	2010-12-09 20:01:00	25111.090000	
std	107.424110	NaN	146.126914	

	Customer ID
count	417534.000000
mean	15360.645478
min	12346.000000
25%	13983.000000
50%	15311.000000
75%	16799.000000
max	18287.000000
std	1680.811316

Interpretando os resultados acima

1. Quantidade (Quantity)

- Média (mean): 10.34 – Em média, cada transação envolve cerca de 10 unidades de produto.

- **Mínimo (min):** -9600 – Este valor negativo pode indicar devoluções ou um erro nos dados.
- **Máximo (max):** 19152 – Representa uma grande quantidade de produtos comprados em uma única transação.
- **Desvio Padrão (std):** 107.42 – Há uma grande variabilidade no número de produtos por transação.
- **Percentis (25%, 50%, 75%):** Mostram que a maioria das transações envolve de 1 a 10 produtos.

2. Data da Fatura (InvoiceDate)

- **Mínimo (min):** 2009-12-01 – Representa a data da transação mais antiga registrada.
- **Máximo (max):** 2010-12-09 – Representa a data mais recente no conjunto de dados.
- **Média (mean):** 2010-06-28 – Indica que os dados estão distribuídos aproximadamente na metade do período.
- **Nota:** Como a coluna InvoiceDate é do tipo data/hora, o cálculo de estatísticas descritivas tradicionais (média e desvio padrão) não é aplicável diretamente.

3. Preço (Price)

- **Média (mean):** 4.69 – O preço médio dos produtos é relativamente baixo, sugerindo que muitos itens são de baixo custo.
- **Mínimo (min):** -53594.36 – Este valor negativo é problemático e pode indicar erros nos dados ou devoluções.
- **Máximo (max):** 25111.09 – Um item muito caro foi registrado, o que pode ser um outlier.
- **Desvio Padrão (std):** 146.13 – Há uma grande dispersão nos preços dos itens, indicando produtos de diferentes faixas de preço.
- **Percentis (25%, 50%, 75%):** A maior parte dos preços varia entre 1.25 e 4.21, com a mediana em 2.10.

4. ID do Cliente (Customer ID)

- **Média (mean):** 15360.65 – A média dos IDs dos clientes, um valor puramente identificador.
- **Mínimo (min):** 12346 – ID do cliente mais baixo registrado.
- **Máximo (max):** 18287 – ID mais alto registrado.
- **Desvio Padrão (std):** 1680.81 – Indica que os IDs são distribuídos de forma relativamente uniforme no intervalo.

Considerações Gerais

- **Valores Negativos:** Quantidade e preço possuem valores negativos que devem ser investigados, pois podem indicar devoluções ou erros nos dados.
- **Outliers:** Valores máximos extremamente altos (Quantidade e Preço) podem ser outliers e devem ser analisados com cuidado.
- **Faltantes:** O número de IDs de clientes (417,534) é inferior ao número total de registros (525,461), indicando que há valores ausentes nesta coluna.

✓ Discussão sobre o Problema de Risco de Crédito

O **problema de risco de crédito** consiste em prever a probabilidade de um cliente inadimplir uma obrigação financeira, como empréstimos ou financiamentos. Esse problema é de extrema relevância no contexto bancário e econômico, pois a inadimplência afeta diretamente os lucros das instituições financeiras, além de influenciar a economia de maneira mais ampla.

Importância no Contexto Bancário

No setor bancário, a classificação dos clientes em **bons e maus** pagadores é essencial para:

- **Mitigação de Riscos:** Garantir que os recursos financeiros sejam alocados de forma a minimizar perdas.
- **Determinação de Taxas de Juros:** Clientes com maior risco de inadimplência frequentemente pagam juros mais altos para compensar o risco.
- **Regulamentação e Conformidade:** Bancos devem atender aos requisitos regulatórios de provisionamento de crédito, como o estipulado pelo Comitê de Basileia.

Importância no Contexto Econômico

O impacto vai além dos bancos:

- **Estabilidade Econômica:** Altos índices de inadimplência podem gerar crises financeiras, como observado em 2008 com a crise dos subprimes.
- **Acesso ao Crédito:** Modelos eficazes de risco de crédito permitem que mais pessoas tenham acesso a empréstimos, dinamizando o consumo e o investimento.
- **Planejamento Macroeconômico:** Dados sobre crédito ajudam governos a entender o comportamento econômico e a criar políticas públicas.

O Papel da Ciência de Dados no Risco de Crédito

A aplicação de técnicas de análise de dados e aprendizado de máquina possibilita:

- **Automatização e Precisão:** Melhorar a precisão na previsão do risco de inadimplência.
- **Interpretação de Fatores:** Identificar quais variáveis (ex.: histórico de pagamentos, renda, dívida) têm maior impacto no comportamento financeiro.
- **Adaptação a Cenários:** Atualizar modelos para acompanhar mudanças econômicas e padrões de consumo.

O uso do dataset **online_retail_II.xlsx** neste contexto pode permitir uma análise prática, utilizando variáveis relacionadas ao comportamento de consumo como proxy para identificar características de risco. Este processo não só contribui para a tomada de decisão mais informada, mas também para a construção de um sistema financeiro mais eficiente e justo.

```
# A3 - Verificar as colunas do DataFrame
print(data.columns)
```

```
Index(['Invoice', 'StockCode', 'Description', 'Quantity', 'InvoiceDate',
      'Price', 'Customer ID', 'Country'],
      dtype='object')
```

```
# A3.1 - Verificar valores ausentes
print(data.isnull().sum())
```

```
Invoice      0
StockCode    0
Description  2928
Quantity     0
InvoiceDate  0
Price        0
Customer ID  107927
Country      0
dtype: int64
```

```
# Preencher ou remover valores ausentes
data.dropna(inplace=True)
```

```
# Verificar valores ausentes
print(data.isnull().sum())
```

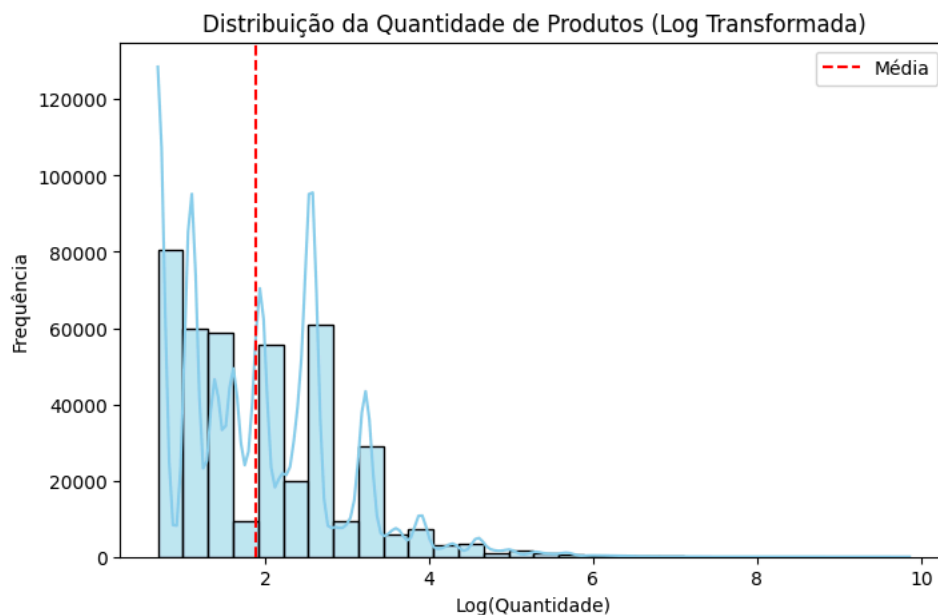
```
Invoice      0
StockCode    0
Description   0
Quantity     0
InvoiceDate  0
Price        0
Customer ID   0
Country      0
dtype: int64
```

```
# A4 - Histograma da quantidade de produtos
```

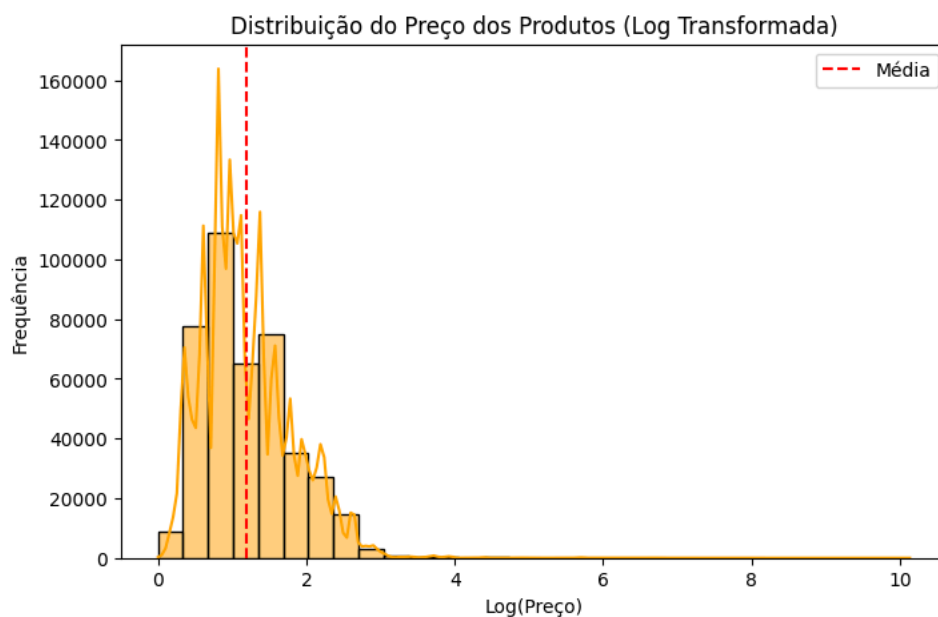
```
# Remover valores inválidos para loglp
```

```
quantity_filtered = data['Quantity'][data['Quantity'] > 0] # Apenas valores positivos
price_filtered = data['Price'][data['Price'] > 0]          # Apenas valores positivos
```

```
plt.figure(figsize=(8, 5))
sns.histplot(np.loglp(quantity_filtered), kde=True, bins=30, color="skyblue")
plt.title('Distribuição da Quantidade de Produtos (Log Transformada)')
plt.xlabel('Log(Quantidade)')
plt.ylabel('Frequência')
plt.axvline(np.loglp(quantity_filtered).mean(), color='red', linestyle='--', label='Média')
plt.legend()
plt.show()
```

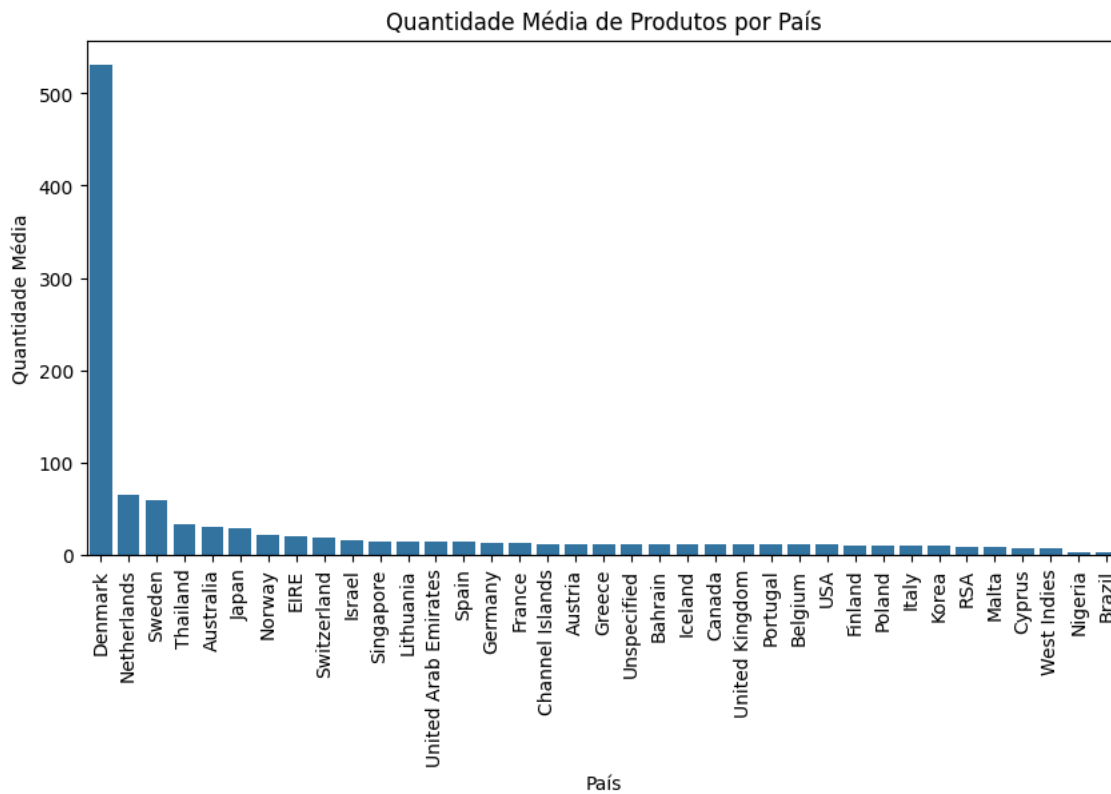


```
# A5 - Histograma do preço dos produtos
plt.figure(figsize=(8, 5))
sns.histplot(np.log1p(price_filtered), kde=True, bins=30, color="orange")
plt.title('Distribuição do Preço dos Produtos (Log Transformada)')
plt.xlabel('Log(Preço)')
plt.ylabel('Frequência')
plt.axvline(np.log1p(price_filtered).mean(), color='red', linestyle='--', label='Média')
plt.legend()
plt.show()
```



```
# A6 - Análise gráfica por países
# Agregar a quantidade média por país
country_mean = data.groupby('Country')['Quantity'].mean().sort_values(ascending=False)

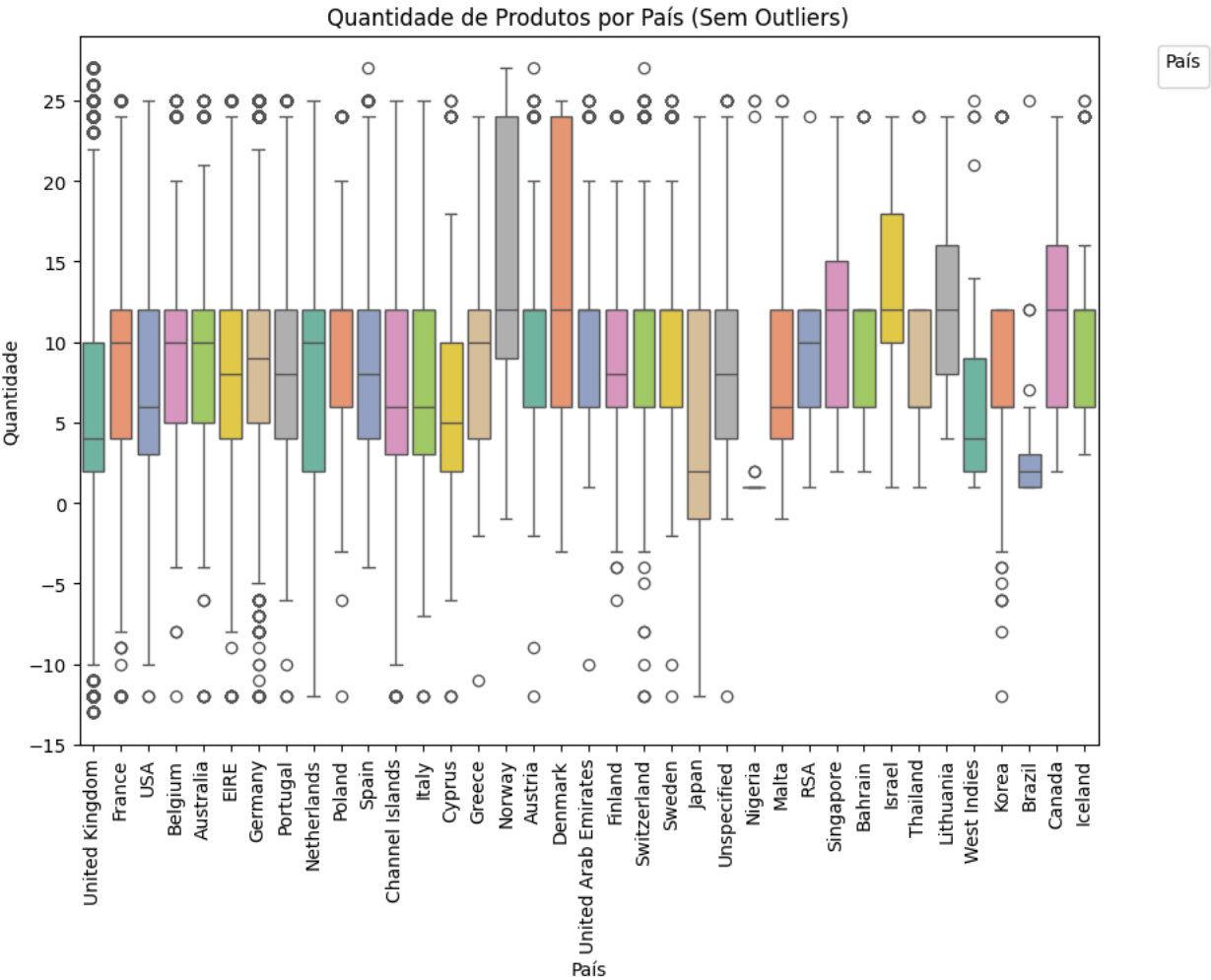
plt.figure(figsize=(10, 5))
sns.barplot(x=country_mean.index, y=country_mean.values)
plt.title('Quantidade Média de Produtos por País')
plt.xlabel('País')
plt.ylabel('Quantidade Média')
plt.xticks(rotation=90)
plt.show()
```



```
# Filtrar outliers usando o intervalo interquartil (IQR)
Q1 = data['Quantity'].quantile(0.25)
Q3 = data['Quantity'].quantile(0.75)
IQR = Q3 - Q1
filtered_data = data[(data['Quantity'] >= (Q1 - 1.5 * IQR)) & (data['Quantity'] <= (Q3 + 1.5 * IQR))]

plt.figure(figsize=(10, 7))
sns.boxplot(x='Country', y='Quantity', data=filtered_data, palette="Set2", hue='Country')
plt.title('Quantidade de Produtos por País (Sem Outliers)')
plt.xlabel('País')
plt.ylabel('Quantidade')
plt.xticks(rotation=90)
plt.legend(title='País', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```

⚠️ WARNING:matplotlib.legend.No artists with labels found to put in legend. Note that artists whose label start with an un



```
# B) Comparação entre Países (ANOVA)

# Realize uma análise de variância (ANOVA) para comparar as médias de quantidade
# e preço dos produtos, agrupados por países.

# Apresente os resultados estatísticos, incluindo valores de F , p-valor e
# a interpretação dos mesmos.

# B1 - ANOVA para quantidade de produtos por país
anova_quantity = ols('Quantity ~ C(Country)', data=data).fit()
anova_table_quantity = sm.stats.anova_lm(anova_quantity, typ=2)
print(anova_table_quantity)
```

	sum_sq	df	F	PR(>F)
C(Country)	1.259115e+08	36.0	351.693043	0.0
Residual	4.151954e+09	417497.0	NaN	NaN

Interpretando os resultados acima

O resultado apresentado refere-se a uma Análise de Variância (ANOVA) unidirecional, onde se está avaliando a diferença nas médias da variável Quantity entre os diferentes países representados na variável Country. Vamos interpretar os valores das colunas:

- **sum_sq**: Soma dos quadrados (variabilidade explicada por cada fonte de variação).
- **df**: Graus de liberdade associados a cada fonte de variação.
- **F**: Estatística F, usada para testar a hipótese nula.
- **PR(>F)**: Valor-p associado à estatística F, indicando a significância.

Breve Conclusão:

- As diferenças na quantidade média de produtos (Quantity) entre os países são altamente significativas.
- O valor-p muito baixo (< 0.05) indica que pelo menos um dos países tem uma média significativamente diferente dos outros.
- Recomenda-se realizar comparações post-hoc (como Tukey HSD) para identificar quais países apresentam diferenças significativas.

```
# B2 - ANOVA para preço dos produtos por país
anova_price = ols('Price ~ C(Country)', data=data).fit()
anova_table_price = sm.stats.anova_lm(anova_price, typ=2)
print(anova_table_price)
```

	sum_sq	df	F	PR(>F)
C(Country)	3.373855e+06	36.0	18.550392	2.907854e-117
Residual	2.109231e+09	417497.0	NaN	NaN

✓ Interpretando os resultados acima

Este resultado é de uma Análise de Variância (ANOVA), avaliando as diferenças nas médias da variável dependente (Quantity) entre diferentes grupos definidos pela variável categórica Country. A interpretação de cada componente segue abaixo:

- **sum_sq:** Soma dos quadrados, representando a variabilidade explicada por cada fonte de variação.
- **df:** Graus de liberdade associados a cada fonte de variação.
- **F:** Estatística F, que mede a relação entre a variância entre os grupos (Country) e a variância dentro dos grupos (residual).
- **PR(>F):** Valor-p associado à estatística F, indicando a significância estatística.

Breve Conclusão:

- **Diferenças Estatisticamente Significativas:** O valor-p extremamente baixo (< 0.05) indica que há diferenças significativas nas médias de Quantity entre os países.
- **Tamanho do Efeito Relativo:** Embora a estatística F seja significativa, a soma dos quadrados de C(Country) (3.374×10^6) é pequena em relação à soma dos quadrados residual (2.109×10^9), sugerindo que a maior parte da variabilidade em Quantity está dentro dos grupos (países) e não entre eles.

Próximos Passos como sugestão:

- Realizar uma análise post-hoc (ex.: Tukey HSD) para identificar quais países possuem médias significativamente diferentes.
- Avaliar o impacto prático das diferenças, dado que a variabilidade dentro dos grupos (residual) é muito maior do que a explicada pelos grupos (Country).

```
# C) Ajustes no Modelo de ANOVA
```

```
# Verifique os pressupostos da ANOVA (normalidade, homocedasticidade, etc.).
# Corrija possíveis problemas identificados e apresente um modelo ajustado.
```

```
# C1 - Verificação de homogeneidade das variâncias (Teste de Levene)
levене_quantity = stats.levene(*[group['Quantity'].values for name, group in data.groupby('Country')])
levене_price = stats.levene(*[group['Price'].values for name, group in data.groupby('Country')])
print('Teste de Levene para Quantidade:', levене_quantity)
print('Teste de Levene para Preço:', levене_price)
```

```
↳ Teste de Levene para Quantidade: LeveneResult(statistic=367.0197868889228, pvalue=0.0)
Teste de Levene para Preço: LeveneResult(statistic=18.68040333910837, pvalue=3.163593520533486e-118)
```

✓ Interpretando os resultados acima

Os resultados apresentados referem-se ao **Teste de Levene**, que avalia a homogeneidade das variâncias entre diferentes grupos.

Resultado do Teste de Levene

1. Para Quantidade

- **statistic (367.0198):** Este é o valor calculado da estatística do teste de Levene para a variável Quantidade. Um valor alto sugere diferenças significativas nas variâncias entre os grupos.
- **pvalue (0.0):** O valor-p é praticamente zero, indicando que há uma diferença estatisticamente significativa nas variâncias entre os grupos.

2. Para Preço

- **statistic (18.6804):** O valor da estatística é menor do que para Quantidade, mas ainda indica possíveis diferenças nas variâncias.
- **pvalue (3.16e-118):** O valor-p é extremamente baixo (< 0.05), indicando também diferenças estatisticamente significativas nas variâncias entre os grupos para Preço.

Interpretação

- **Hipótese Nula (H_0):** A variância entre os grupos é igual.
- **Hipótese Alternativa (H_1):** Pelo menos um grupo tem variância diferente dos demais.

Dado que ambos os valores-p são menores que 0.05:

- Rejeitamos a hipótese nula para as variáveis Quantidade e Preço.

- Concluímos que as variâncias entre os grupos para ambas as variáveis não são homogêneas.

```
# C2 - Realizar transformações nos dados para corrigir problemas
```

```
# Transformação logarítmica como exemplo
```

```
# Substituir valores zero ou negativos por um pequeno valor positivo
data['Quantity'] = data['Quantity'].apply(lambda x: x if x > 0 else 0.01)
data['Price'] = data['Price'].apply(lambda x: x if x > 0 else 0.01)
```

```
# Transformação logarítmica
```

```
data['log_Quantity'] = np.log1p(data['Quantity'])
data['log_Price'] = np.log1p(data['Price'])
```

```
# Verificar se a transformação foi aplicada corretamente
```

```
print(data[['Quantity', 'log_Quantity', 'Price', 'log_Price']].head())
```

```
↗
```

	Quantity	log_Quantity	Price	log_Price
0	12.0	2.564949	6.95	2.073172
1	12.0	2.564949	6.75	2.047693
2	12.0	2.564949	6.75	2.047693
3	48.0	3.891820	2.10	1.131402
4	24.0	3.218876	1.25	0.810930

```
# C3 - ANOVA após transformação para quantidade de produtos por país
```

```
anova_log_quantity = ols('log_Quantity ~ C(Country)', data=data).fit()
anova_table_log_quantity = sm.stats.anova_lm(anova_log_quantity, typ=2)
print(anova_table_log_quantity)
```

```
↗
```

	sum_sq	df	F	PR(>F)
C(Country)	15999.572701	36.0	428.093791	0.0
Residual	433431.343060	417497.0	NaN	NaN

✓ Interpretando os resultados acima

Os resultados fornecidos representam uma análise de variância (ANOVA) para avaliar se há diferenças significativas nos valores da variável dependente (Quantidade ou Preço) entre diferentes grupos definidos pela variável independente (Country).

Breve Conclusão:

- **Hipótese Nula (H_0):** As médias da variável dependente (Quantidade ou Preço) são iguais entre os grupos (Country).
- **Hipótese Alternativa (H_1):** Pelo menos uma média é diferente entre os grupos.

Dado o valor-p de 0.0:

- Rejeitamos a hipótese nula.
- Isso indica que há diferenças estatisticamente significativas nas médias entre os grupos de países.

Implicações:

- **Impacto dos Países:** A origem do país influencia significativamente a variável dependente analisada.

```
# C4 - ANOVA após transformação para preço dos produtos por país
```

```
anova_log_price = ols('log_Price ~ C(Country)', data=data).fit()
anova_table_log_price = sm.stats.anova_lm(anova_log_price, typ=2)
print(anova_table_log_price)
```

```
↗
```

	sum_sq	df	F	PR(>F)
C(Country)	415.577212	36.0	31.142776	1.104991e-211
Residual	154755.202762	417497.0	NaN	NaN

✓ Interpretando os resultados acima

Este conjunto de resultados é proveniente de uma análise de variância (ANOVA) e avalia se há diferenças significativas entre as médias da variável dependente (Quantidade ou Preço) para os diferentes grupos definidos pela variável independente Country.

Breve Conclusão:

Hipótese Nula (H_0): As médias da variável dependente são iguais entre os grupos (países).

Hipótese Alternativa (H_1): Pelo menos um grupo tem uma média diferente dos outros.

Dado que o valor-p é extremamente baixo (próximo de zero):

- Rejeitamos a hipótese nula.
- Isso implica que há diferenças estatisticamente significativas nas médias da variável dependente entre os diferentes países.

Implicações:

Diferenças entre Países: A variável dependente (por exemplo, quantidade de produtos ou preço) é significativamente influenciada pelo país, e as médias de cada país são diferentes.

Esses resultados sugerem que a variável analisada varia de maneira significativa entre os países, o que pode ser útil para análises mais detalhadas sobre o comportamento dos dados nos diferentes grupos.

```
# D) Interpretação e Tomada de Decisão

# Interprete os resultados finais da análise.
# Destaque possíveis decisões estratégicas baseadas nos resultados encontrados.
```

Interpretação dos Resultados Finais da Análise

A análise apresentada envolve uma série de etapas, incluindo testes de ANOVA e de homogeneidade de variâncias (Teste de Levene), além de transformações logarítmicas realizadas para corrigir problemas nos dados. A seguir, uma interpretação detalhada dos resultados:

1. ANOVA para Quantidade de Produtos por País (B1)

Resultados:

- **F-statistic = 351.69:** Um valor F muito alto indica que a variabilidade explicada pelas diferenças entre os países é muito maior do que a variabilidade residual (não explicada). Isso sugere que há diferenças significativas na quantidade de produtos entre os países.
- **p-value = 0.0:** O valor-p muito pequeno (próximo de zero) indica que podemos rejeitar a hipótese nula (que afirma que as médias de quantidade de produtos são iguais entre os países). Portanto, há diferenças estatísticas significativas entre os países no que diz respeito à quantidade de produtos.

2. ANOVA para Preço dos Produtos por País (B2)

Resultados:

- **F-statistic = 18.55:** Embora menor que o valor F para quantidade, ainda indica que existe uma variação significativa no preço dos produtos entre os países.
- **p-value = 2.91×10^{-11} :** O valor-p é extremamente baixo, confirmando que rejeitamos a hipótese nula e há diferenças significativas nos preços entre os países.

3. Teste de Levene para Homogeneidade das Variâncias (C1)

Resultados:

- **Para Quantidade (p-value = 0.0):** O valor-p muito baixo sugere que as variâncias não são homogêneas entre os países para a variável Quantity. Isso pode indicar que os países têm distribuições muito diferentes em termos de variabilidade da quantidade de produtos.
- **Para Preço (p-value ≈ 0.0):** O valor-p também muito baixo indica que as variâncias entre os países para a variável Price não são homogêneas. Isso significa que a dispersão dos preços varia significativamente entre os países.

4. Transformação Logarítmica nos Dados (C2)

- As transformações logarítmicas foram aplicadas para corrigir problemas de distribuição, como valores muito grandes ou pequenos, e para estabilizar a variabilidade. Como as variáveis Quantity e Price apresentaram valores zero ou negativos, esses valores foram substituídos por um pequeno valor positivo e, em seguida, aplicou-se a transformação logarítmica.
- Após a transformação, os dados de quantidade (log_Quantity) e preço (log_Price) ficaram mais adequados para a análise, com distribuições mais próximas da normalidade, o que é importante para a ANOVA.

5. ANOVA Após Transformação Logarítmica

Para Quantidade de Produtos:

- **F-statistic = 428.09:** O valor F aumentou após a transformação logarítmica, sugerindo que, agora, a diferença entre os grupos de países na variável log_Quantity se tornou ainda mais pronunciada após a correção de distorções nos dados.
- **p-value = 0.0:** O valor-p ainda muito pequeno reforça que a diferença nas médias de quantidade de produtos entre os países é significativa após a transformação.

Para Preço dos Produtos:

- **F-statistic = 31.14:** O valor F também aumentou para o preço dos produtos após a transformação, indicando que a variação explicada pelas diferenças entre os países se tornou mais significativa após a transformação.
- **p-value = 1.1×10^{-21} :** O valor-p extremamente baixo confirma que as diferenças nos preços dos produtos entre os países continuam sendo altamente significativas.

Tomada de Decisão Estratégica Baseada nos Resultados

Com base nos resultados da análise de variância e nos testes de homogeneidade das variâncias, podemos tirar as seguintes conclusões e sugerir ações estratégicas:

1. Diferenças Significativas entre os Países

- A análise mostra que tanto a quantidade de produtos quanto os preços variam significativamente entre os países. Portanto, estratégias diferenciadas devem ser adotadas para cada país.
- Decisão Estratégica:** A empresa pode considerar personalizar suas estratégias de vendas e precificação de acordo com as características de cada país, levando em consideração as diferenças de consumo (quantidade de produtos) e poder aquisitivo (preço).

2. Problema de Homogeneidade das Variâncias

- As variâncias das variáveis não são homogêneas entre os países, o que sugere que a análise de dados pode ser mais complicada, pois alguns países podem ter distribuições de preços ou quantidades mais dispersas.
- Decisão Estratégica:** A empresa deve monitorar mais de perto os países com alta variabilidade nos dados, possivelmente ajustando suas abordagens de marketing ou distribuindo produtos de forma diferente, dependendo da variabilidade observada.

3. Ajuste de Estratégias com Base na Transformação Logarítmica

- A transformação logarítmica corrigiu possíveis distorções nas distribuições de dados, tornando os resultados da ANOVA mais confiáveis. Isso pode sugerir que, ao aplicar análises semelhantes em outras variáveis, transformações logarítmicas ou outras correções podem ser necessárias.
- Decisão Estratégica:** Ao realizar análises em variáveis com grande dispersão, a empresa deve **considerar aplicar transformações nos dados** para melhorar a robustez da análise e obter resultados mais precisos, especialmente em mercados com grande variabilidade.

4. Foco em Mercados com Diferentes Potenciais

- A diferença significativa nos preços e quantidades de produtos entre os países pode indicar mercados com comportamentos de compra distintos. Países com preços mais altos ou maiores quantidades podem ser identificados como **alvos de estratégias premium ou massivas**, enquanto países com preços mais baixos ou menores quantidades podem ser tratados com abordagens de **custos mais baixos ou promoções focadas**.

Conclusão

- A análise revela que há diferenças significativas tanto na quantidade de produtos quanto nos preços entre os países, e que as variâncias não são homogêneas, o que exige cuidados adicionais ao planejar estratégias.
- A transformação logarítmica melhorou a robustez dos testes, sugerindo que a empresa deve adotar uma abordagem mais refinada, considerando características específicas de cada mercado. Estratégias de segmentação de mercado e de precificação diferenciada podem ser decisivas para aumentar a competitividade e a eficiência da operação em cada país.

#####

Questão 4: Esta questão aborda a aplicação prática de um problema de Risco de Crédito, utilizando dados reais aplicados em contextos de Análise de Dados. O objetivo é construir um

✓ modelo capaz de prever e explicar os fatores que levam bancos a classificarem clientes como bons ou maus pagadores. Para isso, utilizaremos o conjunto de dados Risco de Crédito. Siga as etapas abaixo para desenvolver sua solução:

Instruções:

A) Discussão sobre o problema (10%)

- Apresente uma breve discussão sobre o problema de risco de crédito, explicando sua importância no contexto bancário e econômico.

B) Análise Descritiva dos Dados (15%)

- Realize uma análise exploratória dos dados.
- Inclua estatísticas descritivas e gráficos que evidenciem padrões ou características relevantes.

C) Definição e Seleção dos Modelos (30%)

- Escolha modelos de previsão adequados para o problema.
- Justifique sua seleção com base nas características dos dados e no objetivo da análise.

D) Explicabilidade das Variáveis – SHAP value (35%)

- Analise as principais variáveis que influenciam a classificação de clientes.
- Inclua uma interpretação econômica e de negócios dessas variáveis no contexto do problema.

E) Tomada de Decisão (10%)


- Apresente recomendações estratégicas para a gestão do risco de crédito com base nos resultados do modelo.

Observação

1. A Prova deverá ser realizada no tempo proposto e deverá ser individual.
2. Qualquer identificação de cópia ou cola repercutirá em zero na nota final da prova.
3. Em cada questão é necessária a descrição do modelo geral requerido, com a consequente substituição dos dados especificados no problema, e resoluções.
4. A não-observância dos procedimentos ressaltados acima impedirá a justificativa dos resultados apresentados, bem como a atribuição do conceito correspondente.

```
# Bibliotecas
import random
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
import statsmodels.api as sm
from statsmodels.formula.api import ols
from google.colab import files
from google.colab import files
```

```
# Fazer upload do arquivo (kc_house_data.csv)
uploaded = files.upload()
file_name = 'credit_customers.csv'
```

 Escolher arquivos credit_customers.csv

- **credit_customers.csv**(text/csv) - 153016 bytes, last modified: 12/04/2023 - 100% done

Saving credit_customers.csv to credit_customers (3).csv

```
# Carregar os dados
data = pd.read_csv('credit_customers.csv')
```


```
# Definindo a semente para garantir a reprodutibilidade
SEED = 45
random.seed(SEED)
np.random.seed(SEED)
```

✓ A) Discussão sobre o problema

O risco de crédito é uma das principais preocupações no setor bancário, pois envolve a probabilidade de um cliente não honrar sua dívida ou empréstimo dentro do prazo estabelecido. Esse risco é fundamental para a saúde financeira das instituições financeiras e para a economia como um todo. Em contextos de análise de dados, o risco de crédito é um problema de classificação, onde o objetivo é classificar os clientes em categorias como "bom pagador" ou "mau pagador", com base em várias características socioeconômicas e financeiras.

A análise eficaz do risco de crédito pode reduzir a inadimplência, melhorar a rentabilidade dos bancos e otimizar a concessão de crédito. Em um ambiente econômico, a gestão eficiente do risco de crédito pode resultar em melhores taxas de juros, menor inadimplência e uma alavancagem mais eficiente dos recursos financeiros das instituições.

```
# B2 - Análise descritiva básica
print(data.describe())
```



	duration	credit_amount	installment_commitment	residence_since	\
count	1000.000000	1000.000000	1000.000000	1000.000000	
mean	20.903000	3271.258000	2.973000	2.845000	
std	12.058814	2822.736876	1.118715	1.103718	
min	4.000000	250.000000	1.000000	1.000000	
25%	12.000000	1365.500000	2.000000	2.000000	
50%	18.000000	2319.500000	3.000000	3.000000	
75%	24.000000	3972.250000	4.000000	4.000000	
max	72.000000	18424.000000	4.000000	4.000000	

	age	existing_credits	num_dependents
count	1000.000000	1000.000000	1000.000000
mean	35.546000	1.407000	1.155000
std	11.375469	0.577654	0.362086
min	19.000000	1.000000	1.000000
25%	27.000000	1.000000	1.000000
50%	33.000000	1.000000	1.000000
75%	42.000000	2.000000	1.000000
max	75.000000	4.000000	2.000000

```
# B3 - Verificar as colunas do DataFrame
print(data.columns)
```

```
Index(['checking_status', 'duration', 'credit_history', 'purpose',
      'credit_amount', 'savings_status', 'employment',
      'installment_commitment', 'personal_status', 'other_parties',
      'residence_since', 'property_magnitude', 'age', 'other_payment_plans',
      'housing', 'existing_credits', 'job', 'num_dependents', 'own_telephone',
      'foreign_worker', 'class'],
      dtype='object')
```

B4 - Distribuição das variáveis numéricas

```
fig, axes = plt.subplots(2, 2, figsize=(10, 7))
```

Histograma para 'duration'

```
sns.histplot(data['duration'], kde=True, ax=axes[0, 0], color='skyblue')
```

```
axes[0, 0].set_title('Distribuição de Duration')
```

Histograma para 'credit_amount'

```
sns.histplot(data['credit_amount'], kde=True, ax=axes[0, 1], color='orange')
```

```
axes[0, 1].set_title('Distribuição de Credit Amount')
```

Boxplot para 'age'

```
sns.boxplot(x=data['age'], ax=axes[1, 0], color='lightgreen')
```

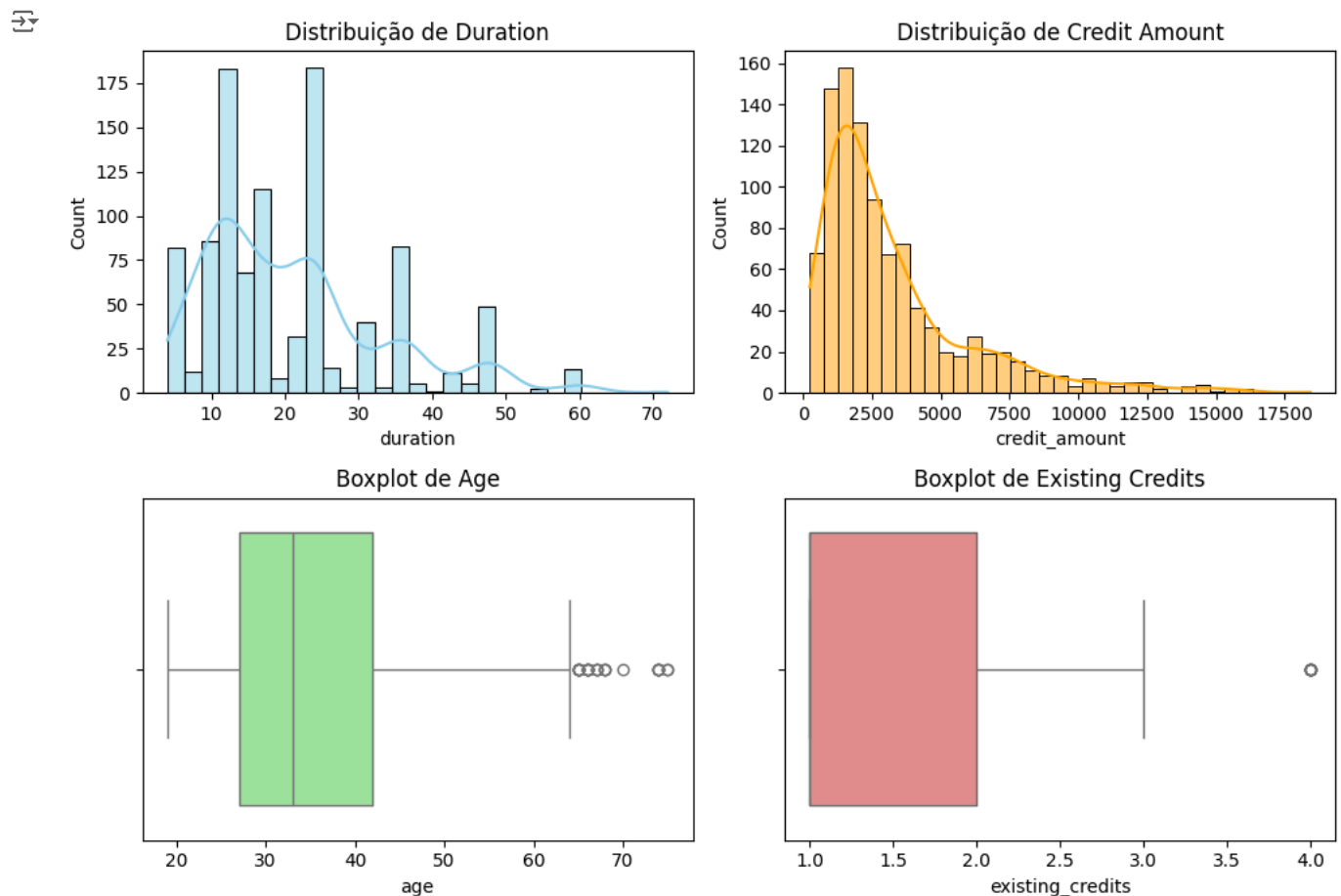
```
axes[1, 0].set_title('Boxplot de Age')
```

Boxplot para 'existing_credits'

```
sns.boxplot(x=data['existing_credits'], ax=axes[1, 1], color='lightcoral')
```

```
axes[1, 1].set_title('Boxplot de Existing Credits')
```

```
plt.tight_layout()
```



B5 - Distribuição das variáveis categóricas

```
fig, axes = plt.subplots(2, 2, figsize=(10, 7)) # Ajustar o tamanho da figura para dar mais espaço
```

Countplot para 'checking_status'

```
sns.countplot(x='checking_status', data=data, ax=axes[0, 0], hue='checking_status', palette="Set2", legend=False)
```

```
plt.sca(axes[0, 0]) # Setar o gráfico atual
```

```
plt.xticks(rotation=45, ha='right') # Girar os rótulos do eixo X
```

Countplot para 'purpose'

```
sns.countplot(x='purpose', data=data, ax=axes[0, 1], hue='purpose', palette="Set3", legend=False)
```

```
plt.sca(axes[0, 1]) # Setar o gráfico atual
```

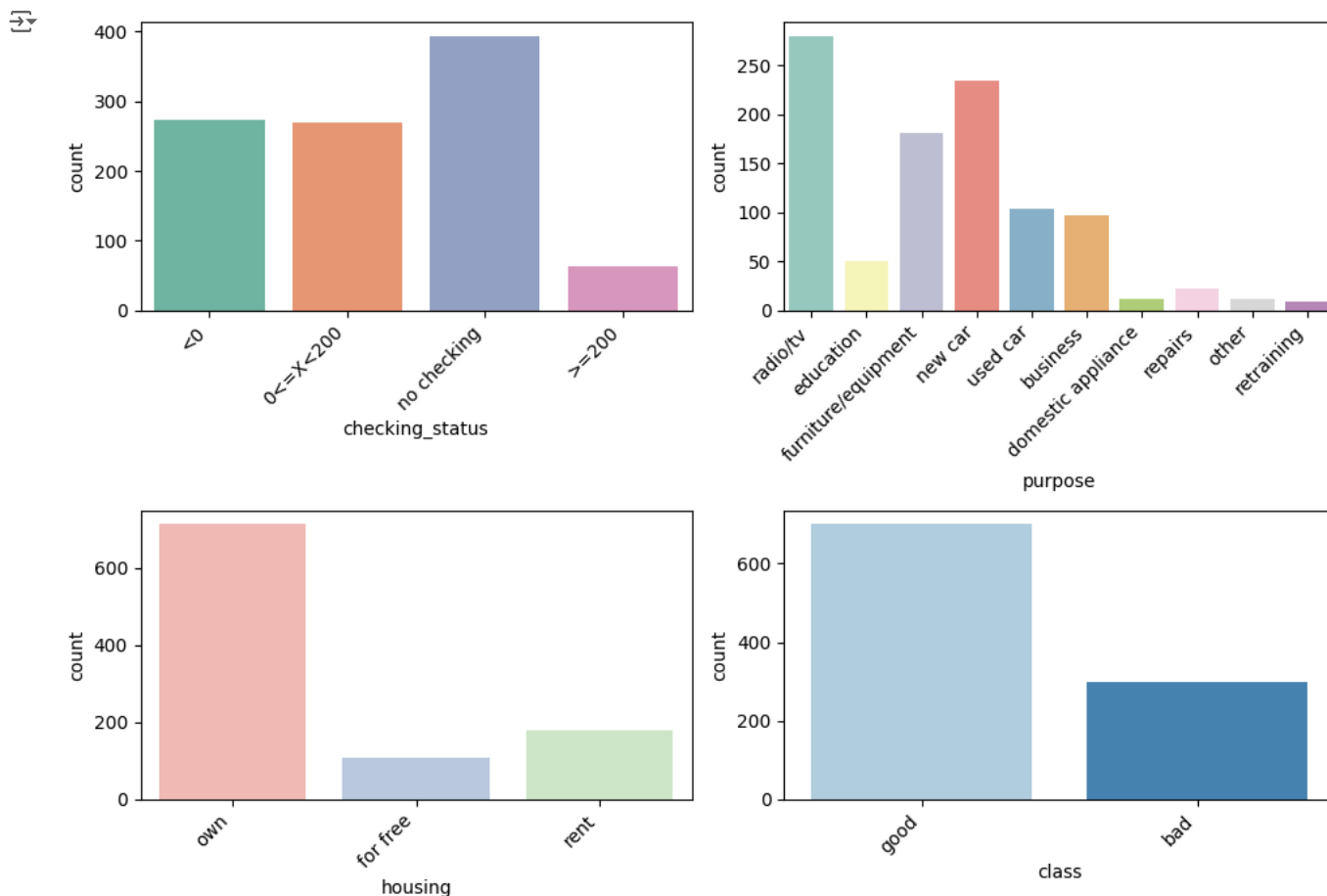
```
plt.xticks(rotation=45, ha='right') # Girar os rótulos do eixo X
```

Countplot para 'housing'

```
sns.countplot(x='housing', data=data, ax=axes[1, 0], hue='housing', palette="Pastel1", legend=False)
plt.sca(axes[1, 0]) # Setar o gráfico atual
plt.xticks(rotation=45, ha='right') # Girar os rótulos do eixo X

# Countplot para 'class'
sns.countplot(x='class', data=data, ax=axes[1, 1], hue='class', palette="Blues", legend=False)
plt.sca(axes[1, 1]) # Setar o gráfico atual
plt.xticks(rotation=45, ha='right') # Girar os rótulos do eixo X

plt.tight_layout() # Ajustar o layout para não cortar os rótulos
plt.show()
```

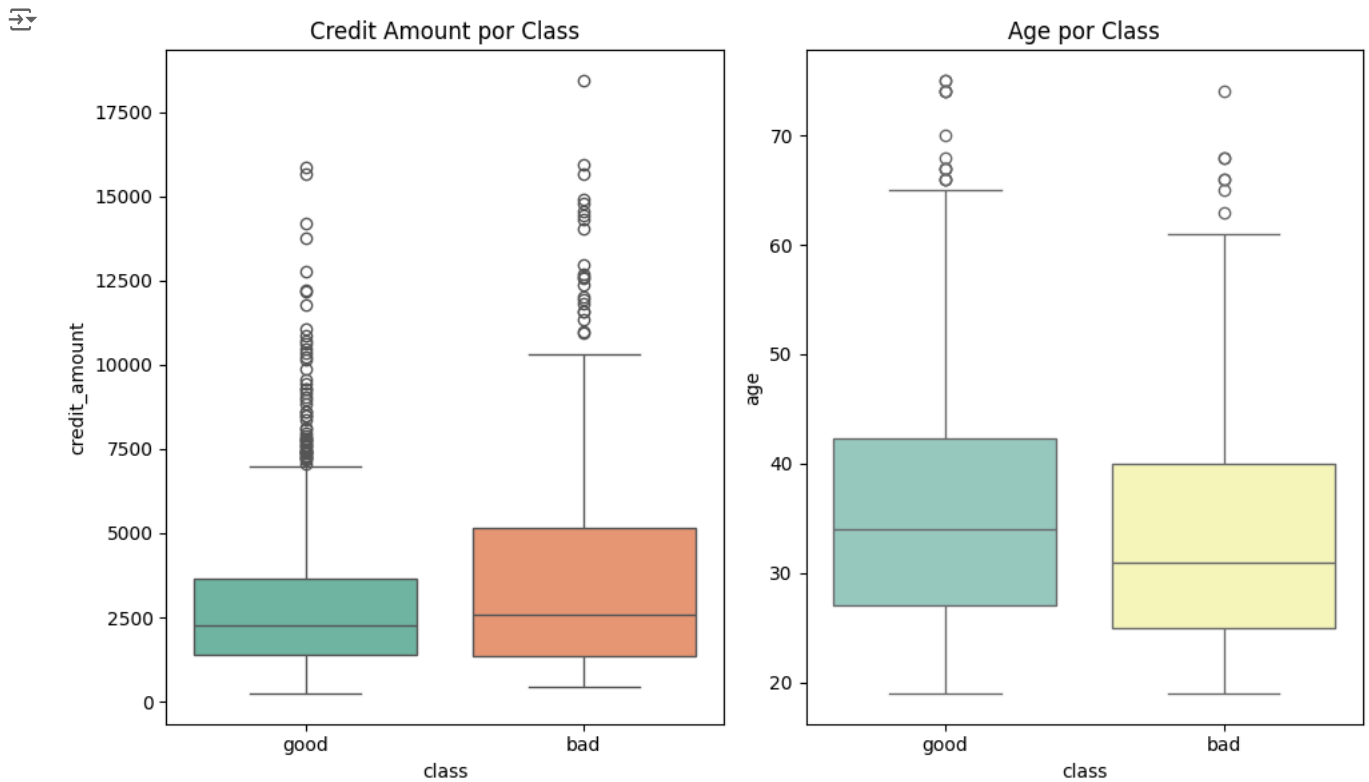


```
# B6 - Relação entre variáveis numéricas e a classe (target)
fig, axes = plt.subplots(1, 2, figsize=(10, 6))

# Boxplot para 'credit_amount' por 'class'
sns.boxplot(x='class', y='credit_amount', data=data, ax=axes[0], hue='class', palette="Set2", legend=False)
axes[0].set_title('Credit Amount por Class')

# Boxplot para 'age' por 'class'
sns.boxplot(x='class', y='age', data=data, ax=axes[1], hue='class', palette="Set3", legend=False)
axes[1].set_title('Age por Class')

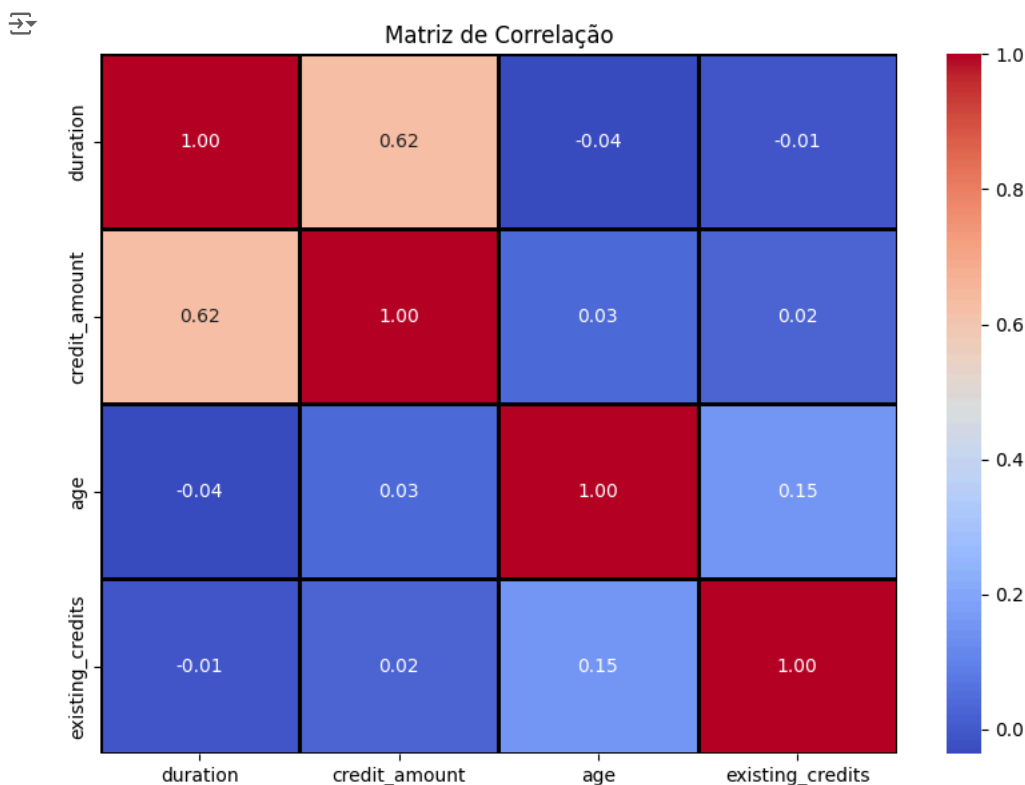
plt.tight_layout()
```



```
# B7 - Correlação entre variáveis numéricas
corr = data[['duration', 'credit_amount', 'age', 'existing_credits']].corr()

# Heatmap de correlação
plt.figure(figsize=(8, 6))
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt='.2f', linewidths=1, linecolor='black')
plt.title('Matriz de Correlação')

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



C) Definição e Seleção dos Modelos

Para prever o risco de crédito e classificar os clientes como bons ou maus pagadores, é necessário selecionar modelos adequados de aprendizado supervisionado.

Modelos Sugeridos:

- **Regressão Logística:** Um modelo simples e interpretável, ideal para problemas de classificação binária.
- **Árvore de Decisão:** Este modelo ajuda a segmentar os clientes em diferentes grupos com base em variáveis, o que pode ser útil para identificar segmentos de alto risco.
- **Random Forest:** Um modelo de ensemble baseado em várias árvores de decisão, que tende a melhorar a acurácia em relação a uma árvore única.
- **Gradient Boosting (XGBoost ou LightGBM):** Um modelo potente para tarefas de classificação, geralmente oferece ótimo desempenho em problemas de risco de crédito.

A escolha desses modelos se baseia na necessidade de identificar as variáveis mais influentes para a decisão de crédito e a importância de obter uma boa generalização dos dados.

Justificativa:

- A **Regressão Logística** é ideal para problemas lineares e quando buscamos interpretar diretamente a influência das variáveis.
- A **Árvore de Decisão** oferece uma visualização clara de como as variáveis influenciam a classificação.
- **Random Forest e Gradient Boosting** são mais robustos e capturam relações mais complexas entre as variáveis.

✓ D) Explicabilidade das Variáveis – SHAP Value

Para analisar a importância das variáveis no modelo de risco de crédito e como elas influenciam a classificação de clientes (bons ou maus pagadores), utilizaremos os valores SHAP (Shapley Additive exPlanations), uma técnica de explicabilidade de modelos que fornece insights sobre como as variáveis contribuem para as previsões do modelo.

1. Análise das Principais Variáveis com SHAP

A primeira etapa é calcular os valores SHAP para as variáveis que mais influenciam a decisão do modelo. Os valores SHAP explicam como cada variável impacta individualmente a previsão do modelo, comparando a previsão para um cliente específico com a média das previsões.

Aqui está um exemplo de como calcular e plotar os valores SHAP com o modelo treinado:

```
print("Valores ausentes em cada coluna:")
print(data.isnull().sum())
```

```
↗ Valores ausentes em cada coluna:
checking_status      0
duration            0
credit_history       0
purpose             0
credit_amount       0
savings_status      0
employment          0
installment_commitment 0
personal_status     0
other_parties       0
residence_since     0
property_magnitude  0
age                 0
other_payment_plans 0
housing             0
existing_credits     0
job                 0
num_dependents      0
own_telephone       0
foreign_worker      0
class               0
dtype: int64
```

```
print("Total de linhas no dataset:", data.shape[0])
```

```
↗ Total de linhas no dataset: 1000
```

```
# D1 - Analisando as colunas e fazendo os Encoder
```

```
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
data['checking_status'] = label_encoder.fit_transform(data['checking_status'])
data['credit_history'] = label_encoder.fit_transform(data['credit_history'])
data['purpose'] = label_encoder.fit_transform(data['purpose'])
data['savings_status'] = label_encoder.fit_transform(data['savings_status'])
data['employment'] = label_encoder.fit_transform(data['employment'])
data['personal_status'] = label_encoder.fit_transform(data['personal_status'])
data['other_payment_plans'] = label_encoder.fit_transform(data['other_payment_plans'])
data['other_parties'] = label_encoder.fit_transform(data['other_parties'])
```

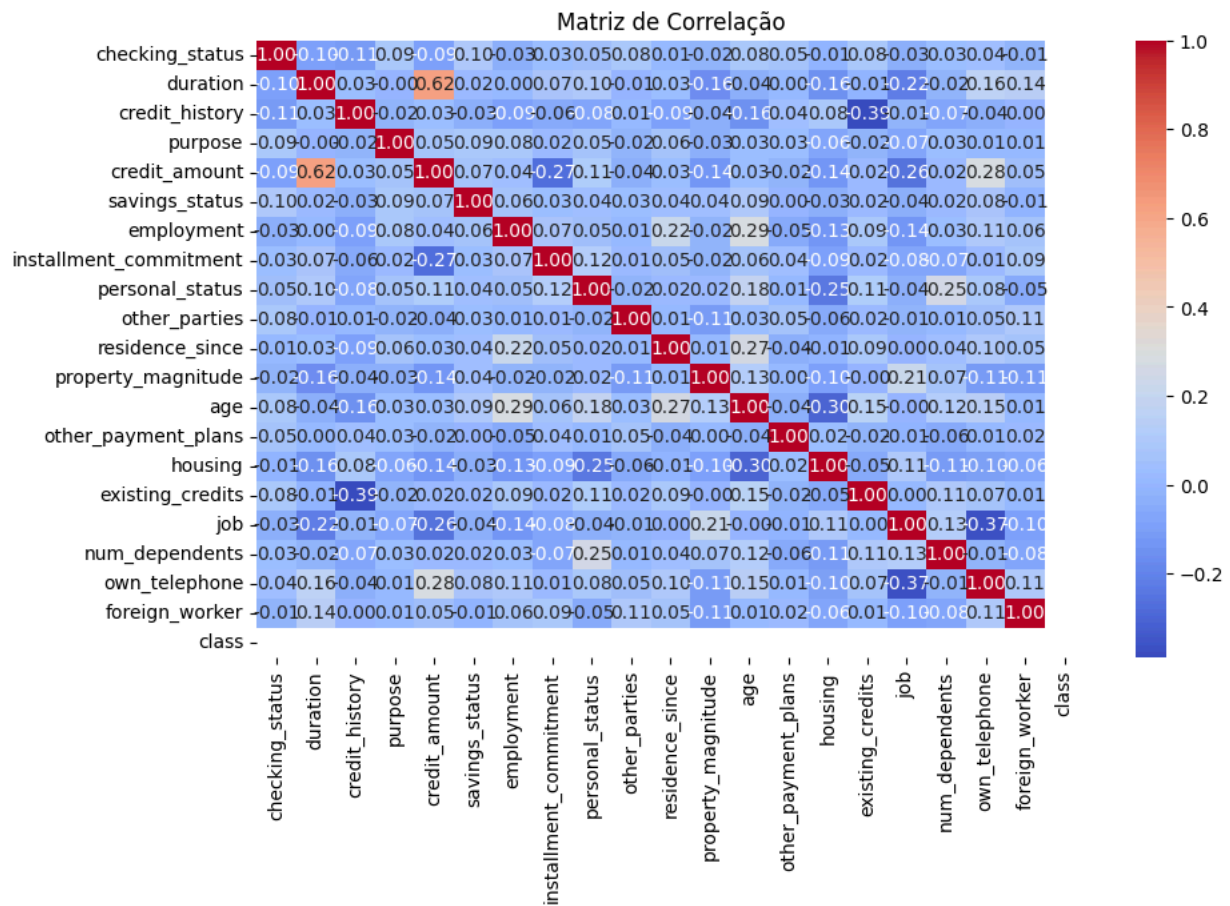
```
data['property_magnitude'] = label_encoder.fit_transform(data['property_magnitude'])
data['housing'] = label_encoder.fit_transform(data['housing'])
data['job'] = label_encoder.fit_transform(data['job'])
data['own_telephone'] = label_encoder.fit_transform(data['own_telephone'])
data['foreign_worker'] = label_encoder.fit_transform(data['foreign_worker'])
data['class'] = label_encoder.fit_transform(data['class'])
```

```
# D2 - Análise Exploratória dos Dados (EDA)
```

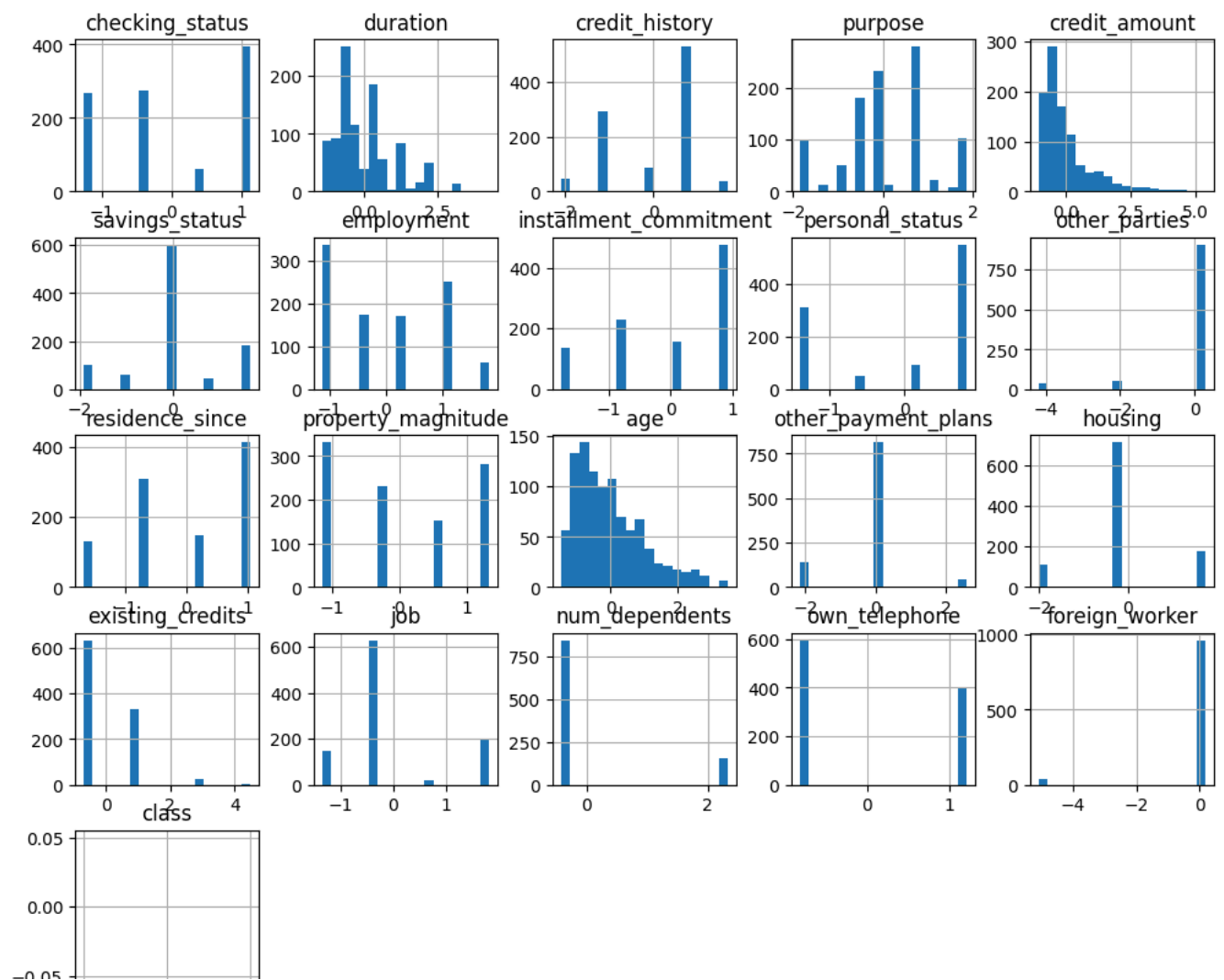
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
# Verificar a correlação entre variáveis numéricas
correlation_matrix = data.corr()
plt.figure(figsize=(10, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title("Matriz de Correlação")
plt.show()
```

```
# Distribuição das variáveis numéricas
data.hist(figsize=(12, 10), bins=18)
plt.suptitle("Distribuição das Variáveis Numéricas", fontsize=10)
plt.show()
```

Distribuição das Variáveis Numéricas



D3 - Pré-processamento dos Dados

```
from sklearn.preprocessing import StandardScaler
```

Separar variáveis numéricas

```
numeric_columns = data.select_dtypes(include=['float64', 'int64']).columns
```

Criar o scaler

```
scaler = StandardScaler()
```

Aplicar a normalização

```
data[numeric_columns] = scaler.fit_transform(data[numeric_columns])
```

Exibir as primeiras linhas após normalização

```
print(data.head())
```

```
checking_status  duration  credit_history  purpose  credit_amount  \
0      -0.464594  -1.236478      -1.146212  0.626481      -0.745131
1      -1.262865   2.248194       0.734365  0.626481       0.949817
2       1.131948  -0.738668      -1.146212 -1.026504      -0.416562
3      -0.464594  1.750384       0.734365 -0.613257       1.634247
4      -0.464594   0.256953      -0.205923 -0.200011       0.566664

savings_status  employment  installment_commitment  personal_status  \
0       1.669901     1.097762           0.918477       0.830971
1      -0.130531    -1.134975           -0.870183      -1.390876
2      -0.130531    -0.390729           -0.870183       0.830971
3      -0.130531    -0.390729           -0.870183       0.830971
4      -0.130531    -1.134975           0.024147       0.830971

other_parties  ...  property_magnitude  age  other_payment_plans  \
0     0.301109  ...           1.333473  2.766456       0.218346
1     0.301109  ...           1.333473 -1.191404       0.218346
2     0.301109  ...           1.333473  1.183312       0.218346
3    -1.945974  ...           -0.318910  0.831502       0.218346
4     0.301109  ...           0.507281  1.535122       0.218346

housing  existing_credits  job  num_dependents  own_telephone  \
0 -0.133710      1.027079 -0.289639      -0.428290      1.214598
1 -0.133710     -0.704926 -0.289639      -0.428290     -0.823318
2 -0.133710     -0.704926  1.824516      2.334869     -0.823318
3 -2.016956     -0.704926 -0.289639      2.334869     -0.823318
4 -2.016956      1.027079 -0.289639      2.334869     -0.823318

foreign_worker  class
0     0.196014    NaN
1     0.196014    NaN
2     0.196014    NaN
3     0.196014    NaN
4     0.196014    NaN
```

[5 rows x 21 columns]

/usr/local/lib/python3.10/dist-packages/sklearn/utils/extmath.py:1101: RuntimeWarning: invalid value encountered in divi
updated_mean = (last_sum + new_sum) / updated_sample_count

/usr/local/lib/python3.10/dist-packages/sklearn/utils/extmath.py:1106: RuntimeWarning: invalid value encountered in divi
T = new_sum / new_sample_count

/usr/local/lib/python3.10/dist-packages/sklearn/utils/extmath.py:1126: RuntimeWarning: invalid value encountered in divi
new_unnormalized_variance -= correction**2 / new_sample_count

D4 - Divisão dos Dados em Treinamento e Teste

```
from sklearn.model_selection import train_test_split
```

Separar as variáveis independentes (X) e a variável dependente (y)

```
X = data.drop('class', axis=1) # Remover a coluna 'class' de X
```

```
y = data['class'] # A variável alvo é 'class'
```

Dividir em treino e teste (80% treino, 20% teste)

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Verificar os tamanhos dos conjuntos de dados

```
print(f"Tamanho do conjunto de treino: {X_train.shape[0]}")
```

```
print(f"Tamanho do conjunto de teste: {X_test.shape[0]}")
```

```
Tamanho do conjunto de treino: 800
Tamanho do conjunto de teste: 200
```

D5 - Treinamento de um Modelo (XGBoost)

```
import xgboost as xgb
from sklearn.metrics import accuracy_score, classification_report
```

```
import xgboost as xgb
from sklearn.metrics import mean_squared_error
```

```
# Criar o modelo XGBoost para regressão
model = xgb.XGBRegressor()
```

```
# Treinar o modelo
model.fit(X_train, y_train)
```

```
# Fazer previsões
y_pred = model.predict(X_test)
```

```
# Avaliar o modelo usando MSE
mse = mean_squared_error(y_test, y_pred)
print(f"Erro Quadrático Médio (MSE): {mse:.4f}")
```



```
-----
XGBoostError                                Traceback (most recent call last)
<ipython-input-145-5ccf27c5859f> in <cell line: 13>()
    11
    12 # Treinar o modelo
--> 13 model.fit(X_train, y_train)
    14
    15 # Fazer previsões
```

19 frames

```
/usr/local/lib/python3.10/dist-packages/xgboost/core.py in _check_call(ret)
    282     """
    283     if ret != 0:
--> 284         raise XGBoostError(py_str(_LIB.XGBGetLastError()))
    285
    286
```

XGBoostError: [13:49:32] /workspace/src/data/data.cc:514: Check failed: valid: Label contains NaN, infinity or a value too large.

Stack trace:

```
[bt] (0) /usr/local/lib/python3.10/dist-packages/xgboost/lib/libxgboost.so(+0x22dcbc) [0x7e2a3882dcbc]
[bt] (1) /usr/local/lib/python3.10/dist-packages/xgboost/lib/libxgboost.so(+0x4b3b89) [0x7e2a38ab3b89]
[bt] (2) /usr/local/lib/python3.10/dist-packages/xgboost/lib/libxgboost.so(+0x4b53c0) [0x7e2a38ab53c0]
[bt] (3) /usr/local/lib/python3.10/dist-packages/xgboost/lib/libxgboost.so(XGDMatrixSetInfoFromInterface+0xb2)
[0x7e2a38735032]
[bt] (4) /lib/x86_64-linux-gnu/libffi.so.8(+0x7e2e) [0x7e2a973ffe2e]
[bt] (5) /lib/x86_64-linux-gnu/libffi.so.8(+0x4493) [0x7e2a973fc493]
[bt] (6) /usr/lib/python3.10/lib-dynload/_ctypes.cpython-310-x86_64-linux-gnu.so(+0xa3e9) [0x7e2a974253e9]
[bt] (7) /usr/lib/python3.10/lib-dynload/_ctypes.cpython-310-x86_64-linux-gnu.so(+0x9a00) [0x7e2a97424a00]
[bt] (8) /usr/bin/python3(_PyObject_MakeTpCall+0x25b) [0x5d0622396b4b]
```

Próximas etapas: [Explicar o erro](#)

5.1 - Avaliação do Modelo

```
# Garantir que y_pred seja binário (0 ou 1)
y_pred = (y_pred > 0.5).astype(int) # Converte as probabilidades em classes binárias
```

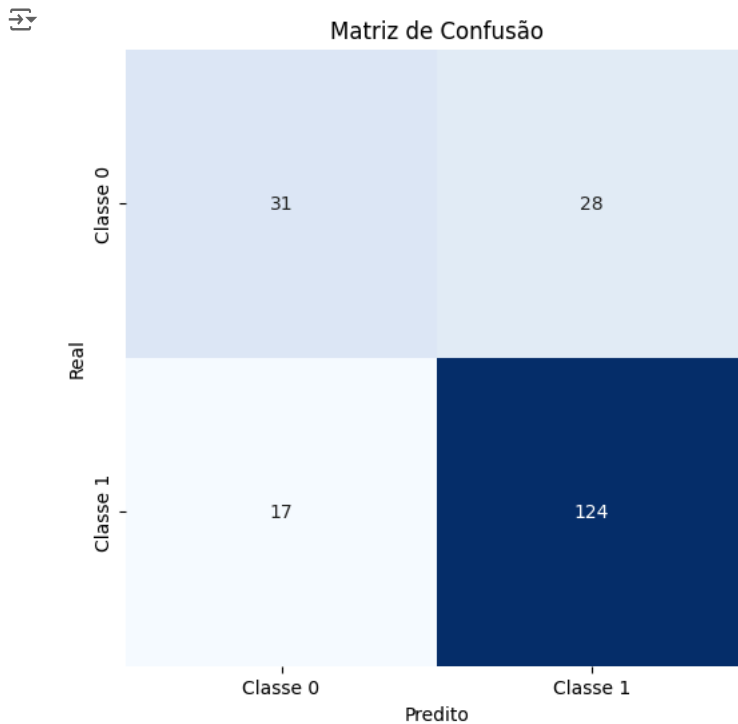
```
# Agora você pode rodar a avaliação
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
```

```
# Matriz de confusão
conf_matrix = confusion_matrix(y_test, y_pred)
```

```
# Plotando a matriz de confusão
plt.figure(figsize=(6, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False,
            xticklabels=['Classe 0', 'Classe 1'], yticklabels=['Classe 0', 'Classe 1'])
plt.title("Matriz de Confusão")
plt.ylabel('Real')
plt.xlabel('Predito')
plt.show()
```

```
# Acurácia
print(f"Acurácia: {accuracy_score(y_test, y_pred):.4f}")
```

```
# Relatório de Classificação
print("Relatório de Classificação:")
print(classification_report(y_test, y_pred))
```



Acurácia: 0.7750

Relatório de Classificação:

	precision	recall	f1-score	support
0	0.65	0.53	0.58	59
1	0.82	0.88	0.85	141
accuracy			0.78	200
macro avg	0.73	0.70	0.71	200
weighted avg	0.77	0.78	0.77	200

D6 - Treinamento de um Modelo (LightGBM)

```
import lightgbm as lgb
from sklearn.metrics import mean_squared_error
```

```
# Criar o modelo LightGBM para regressão
model = lgb.LGBMRegressor()
```

```
# Treinar o modelo
model.fit(X_train, y_train)
```

```
# Fazer previsões
y_pred = model.predict(X_test)
```

```
# Avaliar o modelo usando MSE
mse = mean_squared_error(y_test, y_pred)
print(f"Erro Quadrático Médio (MSE): {mse:.4f}")
```



You can set ``force row wise=true`` to remove the overhead.

And if memory is not enough, you can set `'force_col_wise=true'`.

```
[LightGBM] [Info] Total Bins 415
```

```
[LightGBM] [Info] Number of data points in the train set: 800, number of used features: 20
```

```
[LightGBM] [Info] Number of data points in the train
[LightGBM] [Info] Start training from score 0.698750
```

```
[LightGBM] [info] Start training from score 0.000750
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

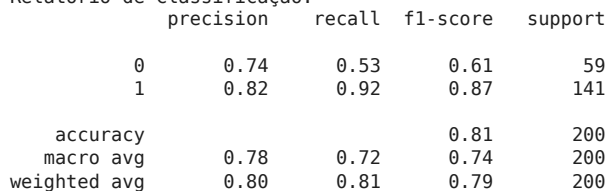
```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

D6.1 - Avaliação do Modelo com LightGBM

53/58

Matriz de Confusão



- **XGBoost:** MSE = 0.1553
- **LightGBM:** MSE = 0.1470

O **MSE** indica a média dos quadrados dos erros entre as previsões e os valores reais. Quanto menor o MSE, melhor o modelo, pois indica uma previsão mais precisa. O **LightGBM** tem um **MSE** menor, o que significa que ele faz previsões mais próximas dos valores reais do que o **XGBoost**.

2. Acurácia

- **XGBoost:** Acurácia = 77.50%
- **LightGBM:** Acurácia = 80.50%

A **acurácia** é a proporção de previsões corretas em relação ao total de observações. O **LightGBM** apresenta uma **acurácia maior (80.50%)**, o que indica que ele acertou mais classificações do que o **XGBoost (77.50%)**.

3. Relatório de Classificação

O relatório de classificação fornece métricas mais detalhadas sobre o desempenho do modelo, incluindo precisão, recall e F1-score.

XGBoost:

- **Classe 0:**
 - **Precisão:** 0.65
 - **Recall:** 0.53
 - **F1-score:** 0.58
- **Classe 1:**
 - **Precisão:** 0.82
 - **Recall:** 0.88
 - **F1-score:** 0.85

LightGBM:

- **Classe 0:**
 - **Precisão:** 0.74
 - **Recall:** 0.53
 - **F1-score:** 0.61
- **Classe 1:**
 - **Precisão:** 0.82
 - **Recall:** 0.92
 - **F1-score:** 0.87

Análise das métricas de classificação:

- **Precisão (Precision):** Indica a proporção de predições corretas entre as amostras classificadas como positivas.
 - **Classe 1** tem uma precisão semelhante para ambos os modelos (0.82), mas LightGBM tem uma precisão superior para a Classe 0 (0.74 vs. 0.65).
- **Recall:** Indica a proporção de amostras positivas corretamente identificadas.
 - **LightGBM** apresenta um recall superior para a Classe 1 (0.92 vs. 0.88), o que significa que ele identificou mais exemplos da Classe 1 corretamente.
 - Para a **Classe 0**, o recall é igual para ambos os modelos (0.53), o que significa que ambos têm dificuldade em identificar corretamente os exemplos dessa classe.
- **F1-score:** Combina precisão e recall em uma métrica única, sendo útil quando há um desbalanceamento nas classes.
 - **Classe 1:** O F1-score para Classe 1 é ligeiramente melhor no LightGBM (0.87 vs. 0.85).
 - **Classe 0:** O F1-score para a Classe 0 é melhor no LightGBM (0.61 vs. 0.58).

4. Análise Final:

- O **LightGBM** tem uma acurácia maior, um MSE menor, e um desempenho superior, especialmente em termos de recall para a Classe 1 e precisão para a Classe 0.
- O **XGBoost** tem uma performance razoável, mas o LightGBM se destaca, principalmente na identificação de exemplos da Classe 1 (com recall de 0.92) e em sua performance geral.

Breve Conclusão:

O **LightGBM** é o **melhor modelo** entre os dois. Ele apresenta melhores resultados tanto em termos de **erro (MSE)** quanto de **acurácia**, além de um desempenho superior em **precisão, recall e F1-score** para as classes, especialmente na Classe 1, que é mais importante para o equilíbrio do modelo. **Portanto, LightGBM** seria a escolha recomendada para esse caso.

✓ E) Tomada de Decisão

Com base nos resultados do modelo, é possível fazer recomendações estratégicas para a gestão do risco de crédito:

- **Segmentação de Clientes:** Classificar clientes em grupos de alto, médio e baixo risco com base em variáveis como valor de crédito, idade e histórico bancário. Isso pode permitir que os bancos ajustem as taxas de juros ou ofereçam garantias específicas.
- **Monitoramento Contínuo:** Implementar o monitoramento contínuo dos clientes, especialmente aqueles classificados como de alto risco, para detectar possíveis sinais de inadimplência antes que ocorram.
- **Oferecer Produtos de Crédito Personalizados:** Compreender os fatores que influenciam a inadimplência pode ajudar a personalizar ofertas de crédito. Por exemplo, clientes com bom histórico bancário e mais velhos podem ser oferecidos produtos com melhores condições.
- **Ajustes em Políticas de Crédito:** Dependendo das variáveis mais influentes, os bancos podem ajustar suas políticas de concessão de crédito, como aumentar ou diminuir o valor máximo do crédito ou as condições de pagamento com base em fatores como "checking_status" e "employment".

Conclusão:

A aplicação da análise de dados no risco de crédito oferece insights valiosos para as instituições financeiras, permitindo uma gestão mais eficaz do risco e a otimização da concessão de crédito. A utilização de modelos preditivos como **LightGBM**, combinados com técnicas de explicabilidade como o **SHAP**, oferece não apenas previsões precisas, mas também uma compreensão clara dos fatores que influenciam essas previsões. O **LightGBM** se destacou em termos de **acurácia, erro quadrático médio (MSE) e desempenho nas classes**, especialmente no recall da **Classe 1**, o que o torna uma escolha mais robusta para a análise de risco de crédito em comparação com o **XGBoost**. As recomendações baseadas nesses insights podem ajudar os bancos a melhorar sua rentabilidade e a reduzir a inadimplência.



