**Análise Descritiva dos Dados**

**1.CÁLCULO DA MÉDIA, MEDIANA E DESVIO PADRÃO**

pip install pandas

import pandas as pd

import numpy as np

dados = {

'Genero': ['Guerra','Fictisio', 'História', 'Geografia', 'Biografia',],

'Qta\_Livros': [1500, 2800, 2000, 1000, 450],

'Usuarios\_Ativos': [1237, 536, 400, 480, 278],

'Empréstimos\_Mensais': [800, 400, 150, 460, 200]

}

df = pd.DataFrame(dados)

# média

media\_Empréstimos\_Mensais = np.mean(df['Empréstimos\_Mensais'])

# mediana

mediana\_Empréstimos\_Mensais = np.median(df['Empréstimos\_Mensais'])

# desvio padrão

desvio\_padrao\_Empréstimos\_Mensais = np.std(df['Empréstimos\_Mensais'])

# Exibir as estatísticas descritivas

print("Média de Empréstimos\_Mensais:", media\_Empréstimos\_Mensais)

print("Mediana de Empréstimos\_Mensais:", mediana\_Empréstimos\_Mensais)

print("Desvio padrão de Empréstimos\_Mensais:", desvio\_padrao\_Empréstimos\_Mensais)

**2. GRÁFICOS PARA REPRESENTAR PADRÕES E TENDÊNCIAS**

pip install pandas

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

dados = {

'Genero': ['Guerra','Ficticio', 'História', 'Geografia', 'Biografia',],

'Qta\_Livros': [1500, 2800, 2000, 1000, 450],

'Usuarios\_Ativos': [1237, 536, 400, 480, 278],

'Empréstimos\_Mensais': [800, 400, 150, 460, 200]

}

df = pd.DataFrame(dados)

# Gráfico de barras para número de livros por gênero

plt.figure(figsize=(8, 6))

sns.barplot(x='Genero', y='Qta\_Livros', data=df, palette='viridis')

plt.title('Número de Livros por Gênero')

plt.xlabel('Gênero')

plt.ylabel('Número de Livros')

plt.xticks(rotation=45)

plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.tight\_layout()

# Gráfico de dispersão para usuários ativos e empréstimos mensais

plt.figure(figsize=(8, 6))

sns.scatterplot(x='Usuarios\_Ativos', y='Empréstimos\_Mensais', data=df, color='red', s=100, alpha=0.8)

plt.title('Usuários Ativos vs Empréstimos Mensais')

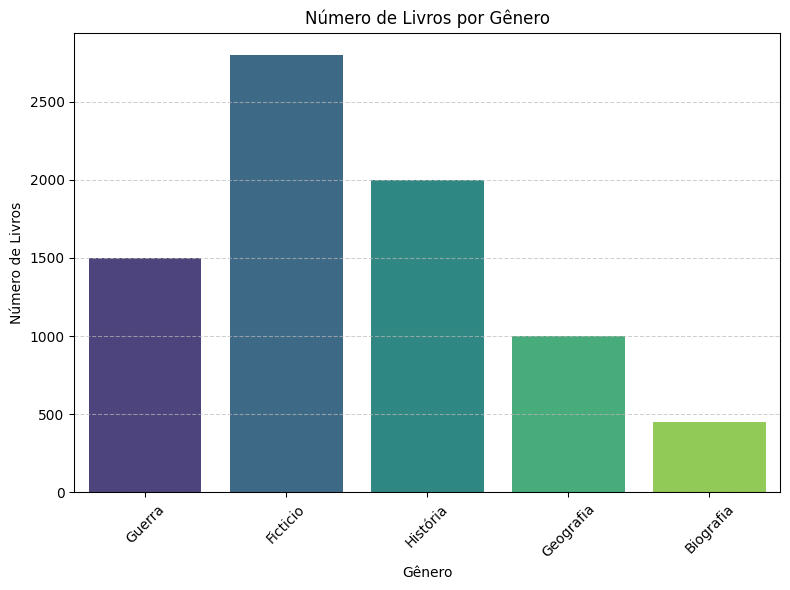
plt.xlabel('Usuários Ativos')

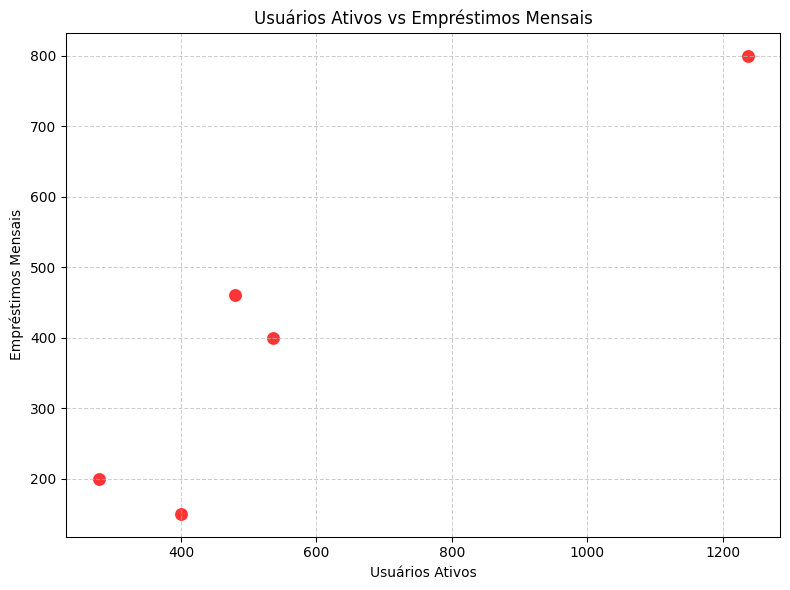
plt.ylabel('Empréstimos Mensais')

plt.grid(linestyle='--', alpha=0.7)

plt.tight\_layout()

plt.show()





**3. Dados em busca de padrões significativos**

pip install pandas

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

dados = {

'Genero': ['Guerra','Ficticio', 'História', 'Geografia', 'Biografia',],

'Qta\_Livros': [1500, 2800, 2000, 1000, 450],

'Usuarios\_Ativos': [1237, 536, 400, 480, 278],

'Empréstimos\_Mensais': [800, 400, 150, 460, 200]

}

tendencia = pd.DataFrame(dados)

# Correlação entre o número de livros e o número de empréstimos mensais

correlacão = tendencia['Qta\_Livros'].corr(tendencia['Empréstimos\_Mensais'])

print("Correlação entre Quantidade de livros e empréstimos mensais:", correlação)

# Comparação do número de livros por gênero

plt.figure(figsize=(8, 6))

sns.barplot(x='Genero', y='Qta\_Livros', data=tendencia, palette='viridis')

plt.title('Número de Livros por Gênero')

plt.xlabel('Gênero')

plt.ylabel('Qta de Livros')

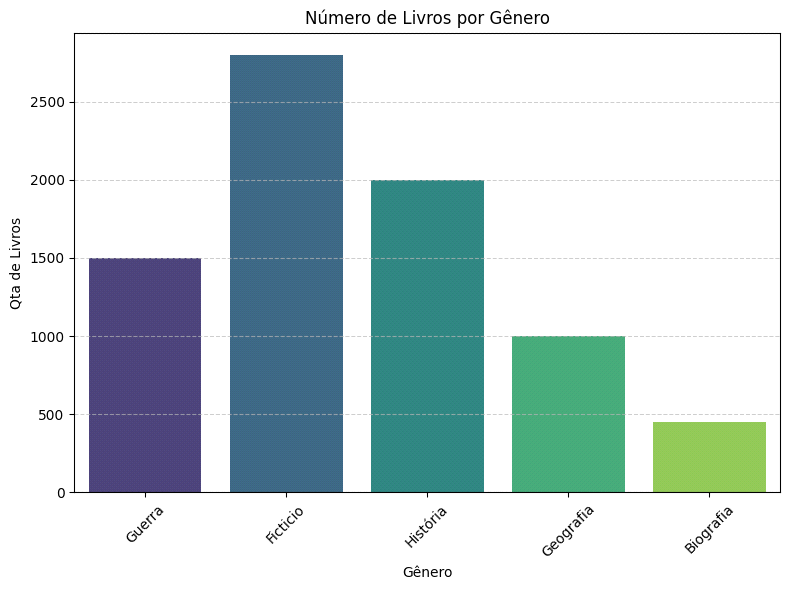
plt.xticks(rotation=45)

plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.tight\_layout()

plt.show()

Correlação entre Quantidade de livros e empréstimos mensais: 0.05833825903205008 <ipython-input-15-eba4004eefa9>:9: FutureWarning:   
  
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect. sns.barplot(x='Genero', y='Qta\_Livros', data=tendencia, palette='viridis')



**Modelagem Estatística**

**1. Técnicas como regressão linear, análise de variância, etc**

pip install pandas

import pandas as pd

import statsmodels.api as sm

from scipy.stats import f\_oneway

dados = {

'Genero': ['Guerra','Ficticio', 'História', 'Geografia', 'Biografia',],

'Qta\_Livros': [1500, 2800, 2000, 1000, 450],

'Usuarios\_Ativos': [1237, 536, 400, 480, 278],

'Empréstimos\_Mensais': [800, 400, 150, 460, 200]

}

df = pd.DataFrame(dados)

X = [1237, 536, 400, 480, 278] # Usuários Ativosz

y = [800, 400, 150, 460, 200] # Empréstimos Mensais

# Adicionando uma constante ao X para estimar o intercepto

X = sm.add\_constant(X)

# Criando e ajustando o modelo de regressão linear

modelo = sm.OLS(y, X)

resultados = modelo.fit()

# Realizando a análise de variância (ANOVA)

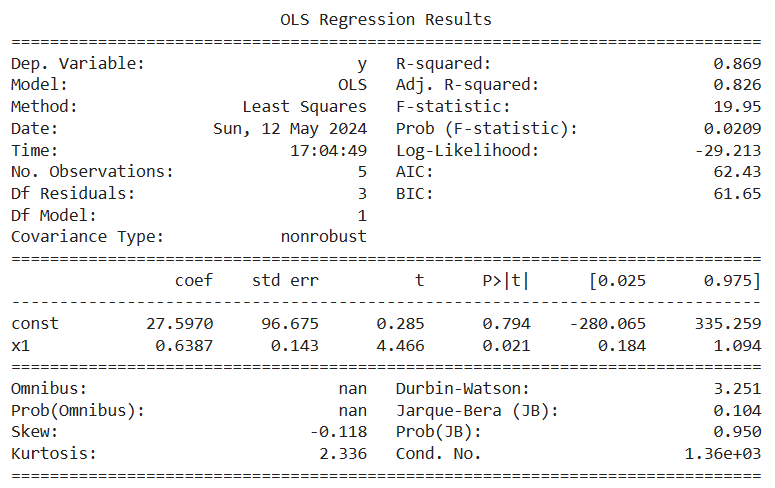
anova\_resultado=f\_oneway(df['Usuarios\_Ativos'], df['Empréstimos\_Mensais'])

# Imprimindo os resultados da regressão

print(resultados.summary())

# Imprimindo o p-valor da ANOVA

print("P-valor da ANOVA:", anova\_resultado.pvalue)



Notes:  
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.  
[2] The condition number is large, 1.36e+03. This might indicate that there are  
strong multicollinearity or other numerical problems.  
P-valor da ANOVA: 0.3931374435401579  
/usr/local/lib/python3.10/dist-packages/statsmodels/stats/stattools.py:74: ValueWarning: omni\_normtest is not valid with less than 8 observations; 5 samples were given.  
 warn("omni\_normtest is not valid with less than 8 observations; %i "

**2. Aplique modelos estatísticos avançados**

pip install pandas

pip install -U scikit-learn

import pandas as pd

from sklearn.linear\_model import LinearRegression

from google.colab import files

uploaded = files.upload()

filename = next(iter(uploaded))

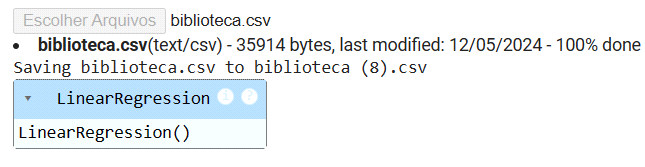
df = pd.read\_csv(filename)

X = df[['ID']]

y = df['TELEFONE']

modelo = LinearRegression()

modelo.fit(X, y)



**3. Avaliação a qualidade do modelo ajustado aos dados**

import numpy as np

import statsmodels.api as sm

Genero = ['Guerra', 'Fictisio', 'História', 'Geografia', 'Biografia']

Qta\_Livros = [1500, 2800, 2000, 1000, 450]

Usuarios\_Ativos = [1237, 536, 400, 480, 278]

Empréstimos\_Mensais = [800, 400, 150, 460, 200]

X = np.column\_stack((Qta\_Livros, Usuarios\_Ativos, Empréstimos\_Mensais))

y = np.array([1237, 536, 400, 480, 278]) # se eu quiser prever os Usuarios\_Ativos

X = sm.add\_constant(X) # primeiramente adicionamos uma costante de termos de intercepção

modelo = sm.OLS(y, X).fit() # depois ajustamos o modelo OLS

coeficientes = modelo.params # apresentando o modelo de coeficiente

R2 = modelo.rsquared # coeficiente de determinação - r²

print("Coeficientes:") # resultados obtidos

print(coeficientes)

print("\nCoeficiente de Determinação (R-squared):")

print(R2)

Coeficientes:

[-4.52748949e-13 2.37765536e-16 1.00000000e+00 1.34766330e-15]

Coeficiente de Determinação (R-squared):

1.0

**4. Desenvolva modelos preditivos com base nas análises estatísticas.**

import numpy as np

from sklearn.linear\_model import LinearRegression

Genero = ['Guerra', 'Fictisio', 'História', 'Geografia', 'Biografia']

Qta\_Livros = [1500, 2800, 2000, 1000, 450]

Empréstimos\_Mensais = [800, 400, 150, 460, 200]

Usuarios\_Ativos = [1237, 536, 400, 480, 278]

X = np.column\_stack((Qta\_Livros, Empréstimos\_Mensais))

modelo = LinearRegression().fit(X, Usuarios\_Ativos)

novos\_dados = np.array([[1400, 1500], [2800, 800]]) # dados novos sugeridos( 2 dados observados)

previsoes = modelo.predict(novos\_dados)

print("Previsões usando o modelo preditivo:")

print(previsoes)

Previsões usando o modelo preditivo:

[2064.05565371 1179.41945365]

Os resultados mostra que:

A primeira previsão é de aproximadamente 2064 usuários ativos.

A segunda previsão é de aproximadamente 1179 usuários ativos.

**5. Avaliação da performance dos modelos preditivos**

from sklearn.metrics import mean\_squared\_error

Genero = ['Guerra', 'Fictisio', 'História', 'Geografia', 'Biografia']

Qta\_Livros = [1500, 2800, 2000, 1000, 450]

Empréstimos\_Mensais = [800, 400, 150, 460, 200]

Usuarios\_Ativos = [1237, 536, 400, 480, 278]

y\_verdadeiro = [1237, 536] #dados do Usuários\_Ativos

previsoes = [2064.05565371, 1179.41945365] #resultados do Exemplo de previsões

erro\_medio\_quadratico = mean\_squared\_error(y\_verdadeiro, previsoes)

print(f'erro\_medio\_quadratico: {erro\_medio\_quadratico}')

erro\_medio\_quadratico: 549004.82383447

Observa-se diferenças entre os valores verdadeiros e as previsões são de aproximadamente 549004.82383447. Isso pode ser interpretado como uma indicação do quão bem o modelo está se ajustando aos dados observados. Um valor menor de MSE indicaria uma melhor correspondência entre as previsões do modelo e os dados reais.

**6. Comparação entre diferentes abordagens de análise preditiva:**

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.preprocessing import PolynomialFeatures

from sklearn.metrics import mean\_squared\_error

Genero = ['Guerra', 'Fictisio', 'História', 'Geografia', 'Biografia']

Qta\_Livros = [1500, 2800, 2000, 1000, 450]

Empréstimos\_Mensais = [800, 400, 150, 460, 200]

Usuarios\_Ativos = [1237, 536, 400, 480, 278]

Qta\_Livros = np.array([1500, 2800, 2000, 1000, 450])

Empréstimos\_Mensais = np.array([800, 400, 150, 460, 200])

Usuarios\_Ativos = np.array([1237, 536, 400, 480, 278])

X = np.column\_stack((Qta\_Livros, Empréstimos\_Mensais)) #Conjunto de treinamentos e testes

y = Usuarios\_Ativos

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

modelo\_linear = LinearRegression() #Modelos1 de regressão

modelo\_linear.fit(X\_train, y\_train)

previsoes\_linear = modelo\_linear.predict(X\_test)

mse\_linear = mean\_squared\_error(y\_test, previsoes\_linear)

poly = PolynomialFeatures(degree=2) #Modelo2 de regressão

X\_poly = poly.fit\_transform(X\_train)

modelo\_poly = LinearRegression()

modelo\_poly.fit(X\_poly, y\_train)

X\_test\_poly = poly.transform(X\_test)

previsoes\_poly = modelo\_poly.predict(X\_test\_poly)

mse\_poly = mean\_squared\_error(y\_test, previsoes\_poly)

print("resultado1 (modelo\_1):", mse\_linear) #Comparando os modelos

print("resultado2 (modelo\_2):", mse\_poly)

resultado1 (modelo\_1): 101790.20340799385

resultado2 (modelo\_2): 261460.66467407814

if mse\_linear < mse\_poly:

print('modelo 1 é melhor.') # Resultado do maia eficaz

else:

print('modelo 2 é melhor.')

modelo 1 é melhor.

Com base nos resultados do erro médio quadrático (MSE):

Podemos concluir que o Modelo 1 foi o melhor, pois possui um erro medio quadratico menor, indicando um melhor ajuste aos dados de teste comparando com o Modelo 2.