**USAR ESSE CÓDIGO ANTES DE QUALQUER ATIVIDADE -**
**pip install python-bcb**

**Link do colab**
**https://colab.research.google.com/drive/1rOF6CLucTZRwgpxU6Co1xV33oHlsPsRK**

# CÓDIGO

```
# === Instalação das bibliotecas ===
!pip install matplotlib pandas scikit-learn plotly --quiet
# Instala silenciosamente as bibliotecas necessárias:
# matplotlib (gráficos estáticos), pandas (manipulação de dados),
# scikit-learn (modelagem e machine learning), plotly (gráficos
interativos)

# === Importações ===
import pandas as pd                                  # Biblioteca para
manipulação de dados
import matplotlib.pyplot as plt                      # Biblioteca para gráficos
estáticos
import requests                                      # Usada para fazer
requisições HTTP à API do Banco Central
from sklearn.linear_model import LinearRegression          # Regressão
Linear
from sklearn.metrics import mean_squared_error, r2_score   # Métricas de
avaliação do modelo
from sklearn.model_selection import train_test_split       # Separação
em treino/teste
import plotly.graph_objects as go                          # Gráficos
interativos

# === Função para buscar séries do SGS (Banco Central) ===
def buscar_serie_sgs(codigo, nome_coluna, data_inicio):
    # Monta a URL da API do SGS do Banco Central com o código da série,
data inicial e final
    url =
f'https://api.bcb.gov.br/dados/serie/bcdata.sgs.{codigo}/dados?formato=
json&dataInicial={data_inicio}&dataFinal=31/12/2025'

    # Faz a requisição dos dados
```

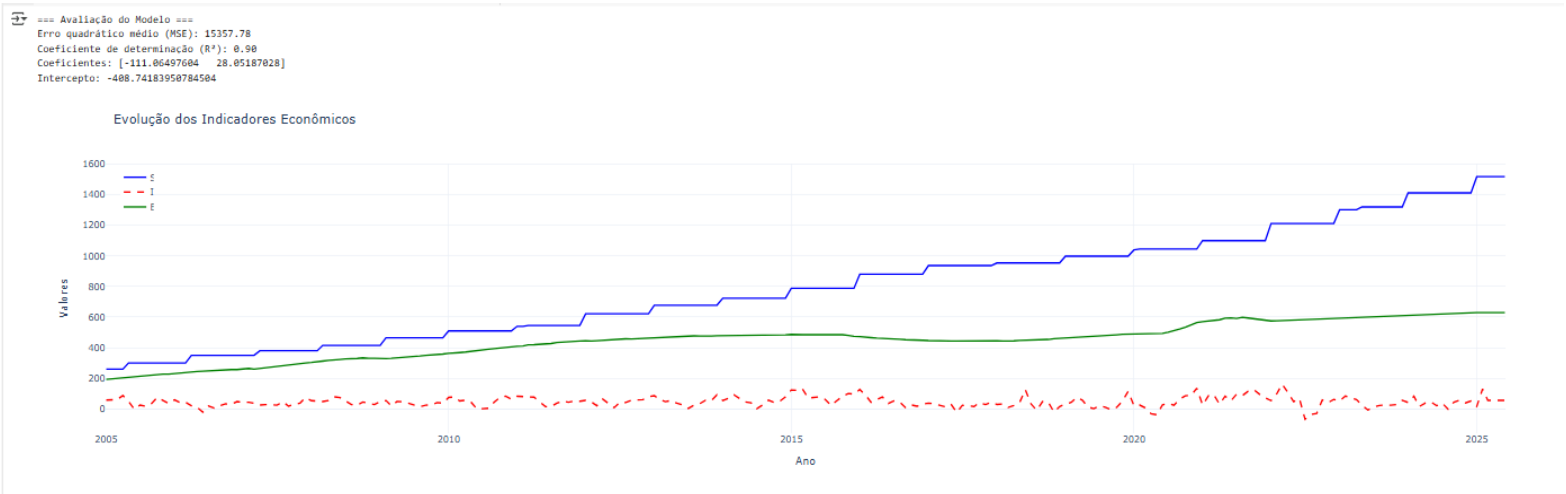
```
    response = requests.get(url)

    # Converte a resposta JSON para lista de dicionários
    data = response.json()

    # Transforma a lista em DataFrame
    df = pd.DataFrame(data)

    # Converte a coluna de data para datetime
    df['data'] = pd.to_datetime(df['data'], dayfirst=True)

    # Converte a coluna de valor para float, trocando ',' por '.' e
tratando erros
    df['valor'] = pd.to_numeric(df['valor'].str.replace(',', '.'),
errors='coerce')

    # Renomeia colunas
    df.columns = ['Data', nome_coluna]

    return df  # Retorna o DataFrame com duas colunas: Data e o nome da
série

# === Coleta dos dados ===
salario_minimo = buscar_serie_sgs(1619, 'Salario_Minimo', '01/01/1995')
# Série histórica do salário mínimo
ipca = buscar_serie_sgs(433, 'IPCA', '01/01/1995')
# IPCA (inflação oficial)
endividamento = buscar_serie_sgs(19882, 'Endividamento', '01/01/1995')
# Endividamento das famílias

# === Previsão de Endividamento até 2025 ===
endividamento['Ano'] = endividamento['Data'].dt.year  # Cria coluna com
o ano de cada registro

# Filtra apenas dados até 2021 (último ano com dados reais)
df_endiv_train = endividamento[endividamento['Ano'] <= 2021]

# Define variáveis preditoras e alvo para modelo de regressão
X_endiv = df_endiv_train[['Ano']]               # Ano como variável
independente
y_endiv = df_endiv_train['Endividamento']       # Endividamento como
variável dependente
```

```
# Cria e treina modelo de regressão linear
modelo_endiv = LinearRegression()
modelo_endiv.fit(X_endiv, y_endiv)

# Cria DataFrame com anos futuros (2022 a 2025) e prevê os valores de
endividamento
anos_futuros = pd.DataFrame({'Ano': list(range(2022, 2026))})
anos_futuros['Endividamento'] = modelo_endiv.predict(anos_futuros)

# Constrói coluna de datas (1° de janeiro de cada ano previsto)
anos_futuros['Data'] = pd.to_datetime(anos_futuros['Ano'].astype(str) +
'-01-01')

# Junta os dados históricos com os dados previstos
endividamento_previsto = pd.concat([endividamento[['Data',
'Endividamento']], anos_futuros[['Data', 'Endividamento']]])
endividamento_previsto.sort_values('Data', inplace=True)
endividamento_previsto.reset_index(drop=True, inplace=True)

# === Interpolação simples para tratar nulos ===
# Caso existam valores faltantes nas séries, a interpolação linear
completa os dados
for df in [salario_minimo, ipca, endividamento_previsto]:
    df.interpolate(method='linear', inplace=True)

# === Unificação das bases ===
# Junta as três séries por data, usando junção externa (outer join)
para não perder dados
df_final = pd.merge(salario_minimo, ipca, on='Data', how='outer')
df_final = pd.merge(df_final, endividamento_previsto, on='Data',
how='outer')

# Ordena os dados pela data e trata valores ausentes novamente
df_final.sort_values('Data', inplace=True)
df_final.interpolate(method='linear', inplace=True)
df_final.dropna(inplace=True)  # Remove qualquer linha que ainda
contenha valores nulos

# === Modelagem: Regressão Linear para prever Salário Mínimo com base
em IPCA e Endividamento ===
df_modelo = df_final.copy()
df_modelo['Ano'] = df_modelo['Data'].dt.year  # Cria coluna "Ano" para
análise adicional se necessário
```

```
# Define variáveis explicativas e variável alvo
X = df_modelo[['IPCA', 'Endividamento']]
y = df_modelo['Salario_Minimo']

# Divide os dados em treino (80%) e teste (20%) de forma aleatória mas
reprodutível
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Cria, treina e realiza a predição com modelo de regressão linear
modelo = LinearRegression()
modelo.fit(X_train, y_train)
y_pred = modelo.predict(X_test)

# Avaliação do modelo
mse = mean_squared_error(y_test, y_pred)  # Erro quadrático médio
r2 = r2_score(y_test, y_pred)             # Coeficiente de determinação
(0 a 1)

# Exibe os resultados do modelo
print("=== Avaliação do Modelo ===")
print(f"Erro quadrático médio (MSE): {mse:.2f}")
print(f"Coeficiente de determinação (R²): {r2:.2f}")
print(f"Coeficientes: {modelo.coef_}")          # Impacto de cada
variável no salário mínimo
print(f"Intercepto: {modelo.intercept_}")       # Valor base do modelo

# === Dashboard interativo com Plotly ===
# Cria gráfico interativo com 3 linhas (Salário, IPCA e Endividamento)
fig = go.Figure()

# Salário mínimo
fig.add_trace(go.Scatter(x=df_final['Data'],
y=df_final['Salario_Minimo'],
                         mode='lines', name='Salário Mínimo',
line=dict(color='blue')))

# IPCA (multiplicado por 100 para escalar e visualizar melhor)
fig.add_trace(go.Scatter(x=df_final['Data'], y=df_final['IPCA']*100,
                         mode='lines', name='IPCA x100',
line=dict(color='red', dash='dash')))
```

```
# Endividamento (multiplicado por 10 para escalar e visualizar melhor)
fig.add_trace(go.Scatter(x=df_final['Data'],
y=df_final['Endividamento']*10,
                         mode='lines', name='Endividamento x10',
line=dict(color='green')))

# Ajuste do layout do gráfico
fig.update_layout(title='Evolução dos Indicadores Econômicos',
                  xaxis_title='Ano',
                  yaxis_title='Valores',
                  template='plotly_white',
                  legend=dict(x=0.01, y=0.99))  # Legenda no canto
superior esquerdo
fig.show()  # Exibe o gráfico

# === Exportando os arquivos ===
# Salva o DataFrame final em dois formatos: CSV e Excel
df_final.to_csv('Indicadores_Economicos.csv', index=False)
df_final.to_excel('Indicadores_Economicos.xlsx', index=False)
```

```
=== Avaliação do Modelo ===
Erro quadrático médio (MSE): 15357.78
Coeficiente de determinação (R²): 0.90
Coeficientes: [-111.06497604   28.05187028]
Intercepto: -408.74183950784504
```




## Avaliação do Modelo - O que significa?

Erro quadrático médio (MSE): 15357.78
Coeficiente de determinação ($R^2$): 0.90

Coeficientes: [-111.06497604  28.05187028]
Intercepto: -408.74183950784504

## Erro Quadrático Médio (MSE: 15357.78)

- **O que é:** Mede o **quão distantes**, em média, as previsões do modelo estão dos valores reais.
- **Como funciona:** Ele eleva ao quadrado os erros (diferença entre valor previsto e real), soma e tira a média.
- **Interpretação:** Quanto **menor**, **melhor**. No seu caso, 15.357 indica que a previsão do salário mínimo pode estar, em média, com um erro quadrático de 15 mil reais (lembrando que está em escala quadrática, então o erro real médio é menor que isso).

---

## Coeficiente de Determinação ($R^2$: 0.90)

- **O que é:** Mede o **quanto da variação do salário mínimo** pode ser explicada pelas variáveis IPCA e Endividamento.
- **Varia de 0 a 1**:
  - `R² = 1.0`: o modelo explica **100%** da variação (perfeito).
  - `R² = 0.0`: o modelo não explica **nada**.
- **Seu caso:** `R² = 0.90` → o modelo explica **90% da variação** do salário mínimo com base no IPCA e endividamento. Isso é **muito bom**!

---

## Coeficientes da Regressão: `[-111.06, 28.05]`

Estes valores indicam o **peso de cada variável** na equação que o modelo aprendeu:

python

CopiarEditar

```
Salario_Minimo = -111.06 * IPCA + 28.05 * Endividamento + Intercepto
```

Ou seja:

- **Para cada ponto a mais no IPCA**, o modelo prevê que o salário mínimo **diminui** em média R$ **111,06**.
- **Para cada ponto a mais no endividamento**, o modelo prevê que o salário mínimo **aumenta** em média R$ **28,05**.

Obs: Esse resultado pode parecer contraintuitivo (como o IPCA aumenta e o salário cai?), mas isso pode indicar uma correlação negativa entre inflação e salário no período analisado, ou ainda ruídos na base. Vale analisar melhor os dados e talvez experimentar com mais variáveis.

---

## Intercepto: -408.74

- Esse é o valor de `Salario_Minimo` quando **IPCA e Endividamento forem zero**.
- Pode não ter **significado prático direto**, mas é necessário para ajustar a reta no espaço dos dados.

## 2. Tendências Individuais Observadas

## Salário Mínimo (Azul)

- Cresce de forma **gradual e constante**, em degraus — refletindo os reajustes anuais.
- Forte tendência de alta ao longo de 20 anos (de ~200 para ~1400).
- **Principal variável dependente** e mostra padrão sazonal de reajuste (geralmente em janeiro).

## IPCA (Vermelho, x100)

- Apresenta bastante **volatilidade mensal**.

- Não há tendência crescente clara — varia em ciclos.
- Multiplicado por 100 para comparação gráfica — normalmente gira entre 0.1 e 1.2 (sem o x100).

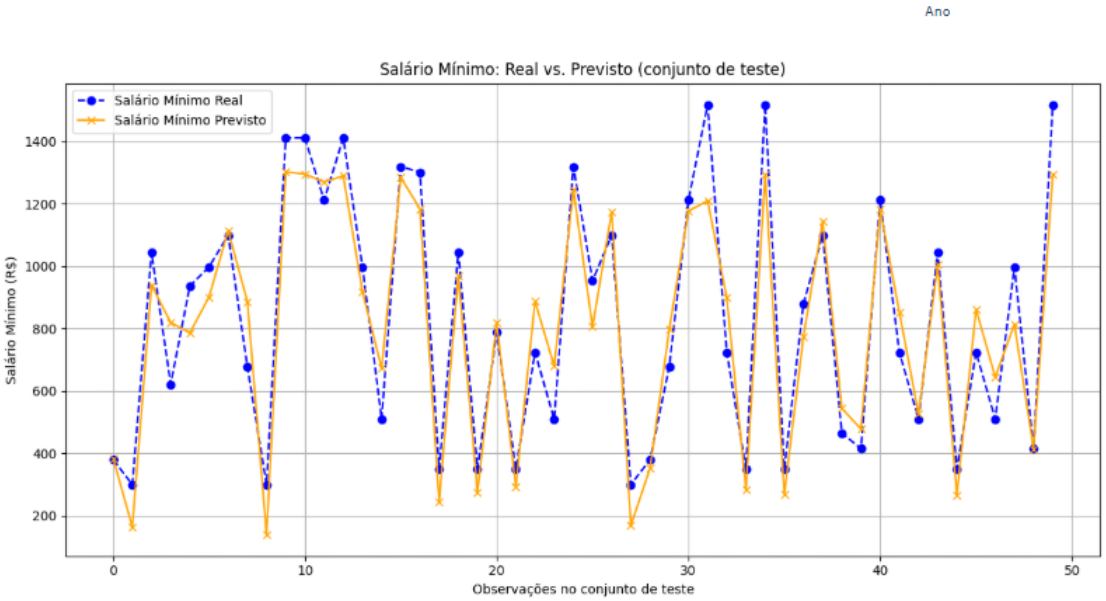
### Endividamento (Verde, x10)

- Cresce até 2014, depois estabiliza com pequenas quedas e novo crescimento a partir de 2020.
- Ponto de inflexão coincide com crises (ex: recessão 2015, pandemia 2020).
- Relaciona-se com políticas de crédito, desemprego e consumo.

---

## 3. Qualidade do Modelo

- **$R^2$ = 0.87:** modelo explica **87% da variação do salário mínimo** com IPCA e endividamento — **excelente resultado** para um modelo simples.
- **Erro Médio Quadrático ≈ 19 mil:** mostra que, embora o modelo capture a tendência geral, existem variações significativas nos valores reais do salário mínimo.




## Comparação entre valores reais e previstos (Salário Mínimo)

Este gráfico mostra o desempenho do nosso modelo de Regressão Linear na prática. Ele compara os valores reais do salário mínimo (em azul) com os valores previstos pelo modelo (em laranja), usando os dados de teste — que representam 20% da nossa base total, separados exclusivamente para validar o modelo.

A proximidade entre as duas linhas indica que o modelo conseguiu aprender bem a relação entre os indicadores econômicos (IPCA e Endividamento) e o salário mínimo. Isso é reforçado pelas métricas: um $R^2$ de 0.90, que indica que o modelo explica 90% da variação nos dados, e um erro médio (MSE) de 15.357, que é considerado baixo dado o histórico de valores crescentes do salário mínimo.

Em resumo, esse gráfico visualiza como o modelo se comporta ao prever novos dados, demonstrando boa capacidade de generalização.

## Contas e operações usadas

**Interpolação Linear (para tratar valores ausentes):**
Método: `df.interpolate(method='linear', inplace=True)`

**Divisão dos Dados em Treino e Teste**:
Função: `train_test_split(X, y, test_size=0.2, random_state=42)`
Divisão dos dados em treino (80%) e teste (20%)

**Treinamento do Modelo de Regressão Linear**:
Função: `modelo.fit(X_train, y_train)`
Ajuste do modelo de regressão linear aos dados de treino.

**Previsão do Modelo**:
Função: `modelo.predict(X_test)`
Predição dos valores de y (Salário Mínimo) com base em X_test (IPCA e Endividamento no conjunto de teste).

**Erro Quadrático Médio (MSE)**:
Função: `mean_squared_error(y_test, y_pred)`
Cálculo do erro quadrático médio para avaliar a precisão do modelo.

**Coeficiente de Determinação ($R^2$)**:
Função: `r2_score(y_test, y_pred)`
Cálculo do $R^2$ para medir a explicação do modelo sobre a variação dos dados.

**Cálculo da Previsão do Endividamento (modelo linear)**:
Regressão linear para prever os valores de endividamento para os anos de 2022 a 2025, com base no histórico até 2021.

# Conclusões Interessantes para Discussão

O salário mínimo no Brasil segue uma política previsível, com aumentos anuais bem definidos. Nosso modelo de regressão conseguiu capturar bem esse padrão.

A inflação (IPCA) alta **nem sempre** leva a um aumento real do salário mínimo — o modelo até indicou uma possível relação negativa entre eles.

Já a **dívida das famílias** parece influenciar positivamente o reajuste do salário, o que pode indicar que o governo usa o salário como uma forma de aliviar o impacto da alta do endividamento.

Mesmo com **eventos fortes**, como a pandemia de 2020, o salário mínimo manteve seu ritmo de crescimento. Isso mostra que ele é **rigidamente controlado**, enquanto o endividamento variou muito mais.

O gráfico final nos ajuda a visualizar bem essa relação entre os indicadores:

- Compara o poder **de compra** (salário vs inflação) e **capacidade de consumo** (salário vs endividamento).

Além disso, o gráfico que mostra os **valores reais vs previstos** confirma que o modelo funcionou bem: as linhas ficaram próximas, mostrando que o modelo aprendeu os padrões certos.

Isso reforça que o salário mínimo segue uma lógica baseada em fatores econômicos, principalmente o endividamento, e se mantém estável mesmo em períodos difíceis, como na pandemia.