

▼ **Prevendo valores do Preço por Barril do Petróleo bruto Brent (FOB)**

O objetivo deste estudo é conduzir quatro tipos de testes de previsão de séries temporais!

Utilizaremos dois modelos tradicionais (média móvel e ARIMA), e também um modelo mais robusto (Prophet do Facebook - Meta)

▼ **Sobre o dataset**

O dataset é composto pelo histórico do valor Preço por barril do petróleo bruto Brent (FOB).  
Produzido no Mar do Norte (Europa), Brent é uma classe de petróleo bruto que serve como benchmark para o preço internacional de diferentes tipos de petróleo. Neste caso, é valorado no chamado preço FOB (free on board), que não inclui despesa de frete e seguro no preço.

Link para acessar os dados de Preço: <http://www.ipeadata.gov.br/ExibeSerie.aspx?module=m&serid=1650971490&oper=view>

▼ **Importando as Bibliotecas**

Clique duas vezes (ou pressione "Enter") para editar


```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.stattools import acf, pacf
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.arima.model import ARIMA
from prophet import Prophet
```

▼ **Construindo o Dataset**

▼ **Importando e checando os dados**


```
# Tentando ler o arquivo CSV com a codificação 'latin1'
dados = pd.read_csv('base_preco_petroleo.csv', sep=';', encoding='latin1')
```

```
# Verificando os dados
dados.head()
```



	Data	Preço do Petróleo
0	13/05/2024	83,18
1	10/05/2024	83,39
2	09/05/2024	83,27
3	08/05/2024	82,44
4	07/05/2024	82,69

```
dados.tail()
```




	Data	Preço do Petróleo
11164	26/05/1987	18,63
11165	25/05/1987	18,6
11166	22/05/1987	18,55
11167	21/05/1987	18,45
11168	20/05/1987	18,63

▼ Mudando o nome das colunas

```
# Renomeando as colunas para facilitar os códigos:
dados.rename(columns={'Data': 'data', 'Preço do Petróleo': 'preco'}, inplace=True)

# Verificando os dados
dados.head()
```




	data	preco
0	13/05/2024	83,18
1	10/05/2024	83,39
2	09/05/2024	83,27
3	08/05/2024	82,44
4	07/05/2024	82,69

▼ Transformando a coluna data em Datetime

```
dados['data'] = pd.to_datetime(dados['data'], format='%d/%m/%Y')

# Verificando os dados
dados.head()
```




	data	preco
0	2024-05-13	83,18
1	2024-05-10	83,39
2	2024-05-09	83,27
3	2024-05-08	82,44
4	2024-05-07	82,69

▼ Transformando a coluna preco em int

```
# Substituindo a vírgula por ponto
dados['preco'] = dados['preco'].str.replace(',', '.')


# verificando a substituição
dados.head()
```



	data	preco
0	2024-05-13	83.18
1	2024-05-10	83.39
2	2024-05-09	83.27
3	2024-05-08	82.44
4	2024-05-07	82.69

```
# transformando preço em numerico
dados['preco'] = pd.to_numeric(dados['preco'])

# verificando se as colunas estão corretas
print(dados.dtypes)
```



data	datetime64[ns]
preco	float64
dtype:	object

▼ Colocando a coluna data como index

```
dados.set_index('data', inplace=True)

# Verificando os dados
```

```
dados.head()
```

```
↗
```

	preco
data	
2024-05-13	83.18
2024-05-10	83.39
2024-05-09	83.27
2024-05-08	82.44
2024-05-07	82.69

```
type(dados) #check para ver se realmente a coluna index
```

```
↗
```

```
pandas.core.frame.DataFrame
def __init__(data=None, index: Axes | None=None, columns: Axes | None=None,
dtype: Dtype | None=None, copy: bool | None=None) -> None

Two-dimensional, size-mutable, potentially heterogeneous tabular data.

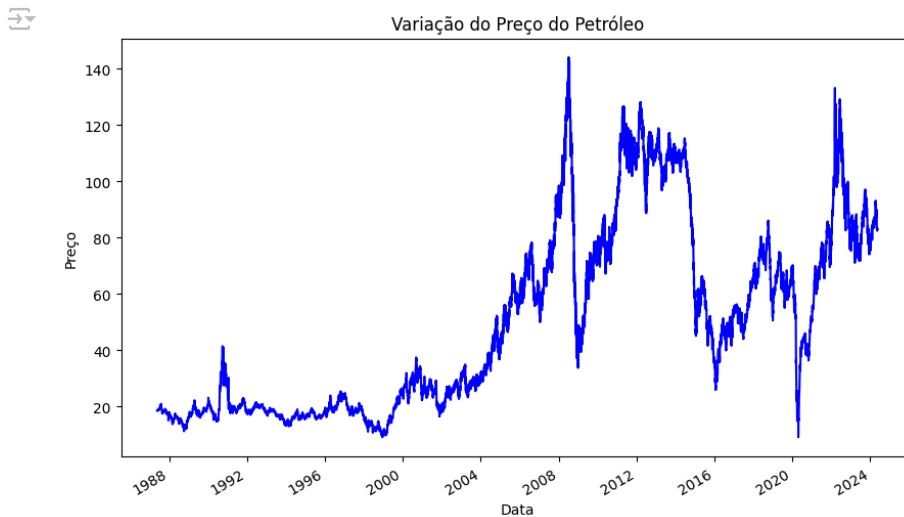
Data structure also contains labeled axes (rows and columns).
Arithmetic operations align on both row and column labels. Can be
thought of as a dict-like container for Series objects. The primary
```

### Visualizando os valores de fechamento das ações

```
dados['preco'].plot(figsize=(10, 6), linestyle='-', color='b')
```

```
# Adicionando rótulos e título ao gráfico
plt.title('Variação do Preço do Petróleo')
plt.xlabel('Data')
plt.ylabel('Preço')
```

```
# Exibindo o gráfico
plt.show()
```



### Decomposição para Análise da série temporal

Utilizaremos o Statsmodel para decompor a série em:

- Tendência, que é a direção da série temporal.
- Sazonalidade, que são os padrões repetidos no tempo.
- Ruído, que é a diferença entre a série original e o explicado por tendência e sazonalidade.

Importância da Decomposição: A decomposição auxilia na compreensão dos dados e na escolha do modelo preditivo, permitindo:

- Previsões de longo prazo ao compreender a tendência.
- Previsões de curto prazo ao entender a sazonalidade.
- Melhora nos modelos ao analisar os resíduos não capturados.

Série Aditiva ou Multiplicativa: A decomposição indica se a série é:

- Aditiva: Magnitude de sazonalidade e tendência constante.
- Multiplicativa: Magnitude varia proporcionalmente ao nível da série.

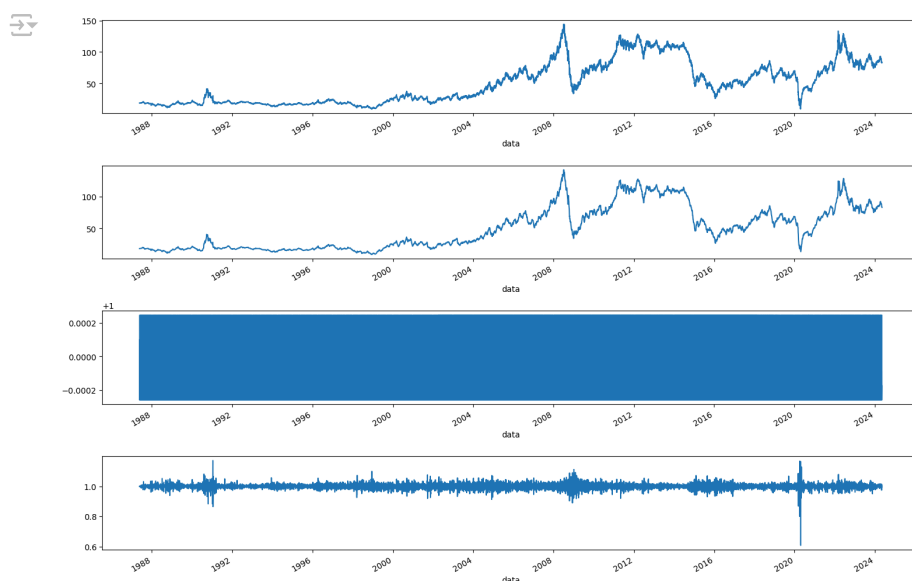
#### ✓ Decompondo todo dataset em serie, tendência, sazonalidade e ruído

```
resultados = seasonal_decompose(dados, model='multiplicative', period=5) # Period = sazonalidade. Como temos dados de segunda a sexta, ent
```

```
fig, (ax1,ax2,ax3,ax4) = plt.subplots(4,1, figsize = (15,10))
```

```
resultados.observed.plot(ax=ax1) # Serie dos dados  
resultados.trend.plot(ax=ax2) # tendência dos dados  
resultados.seasonal.plot(ax=ax3) # sazonalidade  
resultados.resid.plot(ax=ax4) # resíduos
```

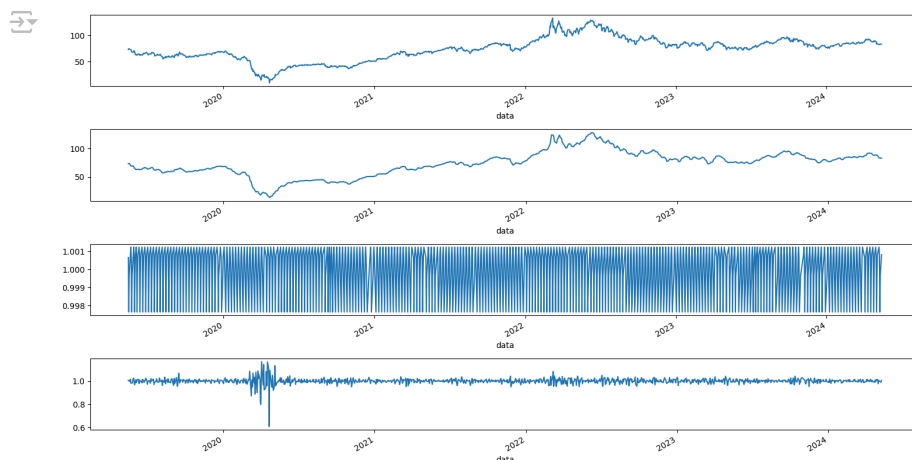
```
plt.tight_layout()
```



#### ✓ Decompondo **5 anos** do dataset em serie, tendência, sazonalidade e ruído

```
df_5_anos = dados.loc['2019-05-15:']
seasonplot = seasonal_decompose(df_5_anos, model='multiplicative', period=5) # decompondo a série temporal

fig, (ax1, ax2, ax3, ax4) = plt.subplots(4, 1, figsize=(15, 8))
seasonplot.observed.plot(ax=ax1) # série real
seasonplot.trend.plot(ax=ax2) # tendência
seasonplot.seasonal.plot(ax=ax3) # sazonalidade
seasonplot.resid.plot(ax=ax4) # resíduos
plt.tight_layout()
plt.show()
```



## ✓ Testando Modelos de Machine Learning Clássicos

### ✓ Média Móvel

A média móvel é uma técnica estatística utilizada na análise de séries temporais e dados financeiros para suavizar flutuações aleatórias e identificar tendências ao longo do tempo. A ideia básica por trás da média móvel é calcular, para cada ponto de dados em uma série temporal, a média dos pontos de dados vizinhos.

Vamos utilizar esta técnica para um N de 10 e de 20 dias.


### ✓ Criando a base

```
dados_MA = dados.copy() # criando uma cópia do DataFrame original

# Adicionando colunas ao DataFrame copiado
dados_MA['MA_window_10'] = dados_MA['preco'].rolling(10).mean().shift() # média móvel em 10 dias
dados_MA['MA_window_20'] = dados_MA['preco'].rolling(20).mean().shift() # média móvel em 20 dias

#Rolling = especifica o número de observações usadas para calcular a estatística
#shift = utilizado para deslocar o índice de DataFrame por um número especificado de períodos com uma frequência de tempo
```

```
dados_MA.head(20)
```



	preco	MA_window_10	MA_window_20
data			
2024-05-13	83.18	NaN	NaN
2024-05-10	83.39	NaN	NaN
2024-05-09	83.27	NaN	NaN
2024-05-08	82.44	NaN	NaN
2024-05-07	82.69	NaN	NaN
2024-05-03	83.60	NaN	NaN
2024-05-02	84.81	NaN	NaN
2024-05-01	83.55	NaN	NaN
2024-04-30	88.23	NaN	NaN
2024-04-29	88.44	NaN	NaN
2024-04-26	89.95	84.360	NaN
2024-04-25	88.10	85.037	NaN
2024-04-24	89.02	85.508	NaN
2024-04-23	88.29	86.083	NaN
2024-04-22	87.30	86.668	NaN
2024-04-19	87.96	87.129	NaN
2024-04-18	88.34	87.565	NaN
2024-04-17	89.54	87.918	NaN
2024-04-16	91.29	88.517	NaN
2024-04-15	90.84	88.823	NaN

▼ Vizualizando o Resultado

```
plt.figure(figsize=(15,8))
plt.grid(True)
plt.plot(dados_MA['preco'], label='Último')
plt.plot(dados_MA['MA_window_10'], label='MA window 10 days')
plt.plot(dados_MA['MA_window_20'], label='MA window 20 days')
plt.legend(loc=2)
plt.show()
```



Visualizando os dados mais recentes: últimos 365 dias

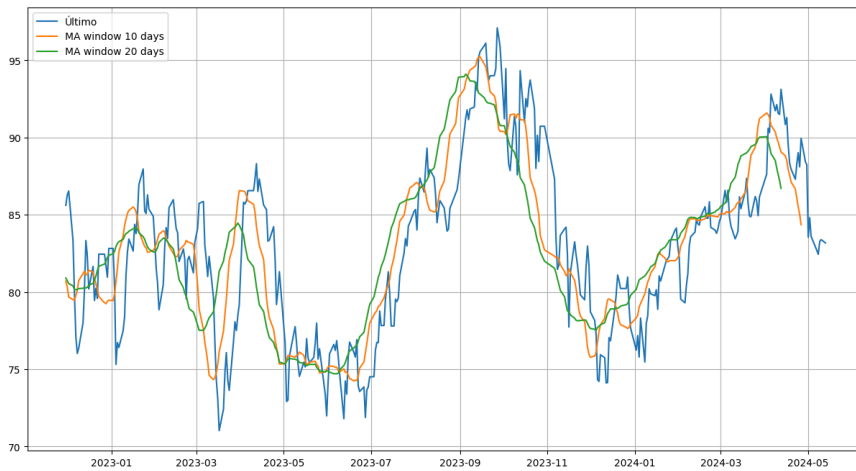
```
import matplotlib.pyplot as plt

# Ordenando o DataFrame pela coluna de datas
dados_MA = dados_MA.sort_index(ascending=True)

# Pegando os últimos 300 dias a partir da data mais recente
df_precos_300_dias = dados_MA.tail(365)

plt.figure(figsize=(15, 8))
plt.grid(True)

plt.plot(df_precos_300_dias['preco'], label='Último')
plt.plot(df_precos_300_dias['MA_window_10'], label='MA window 10 days')
plt.plot(df_precos_300_dias['MA_window_20'], label='MA window 20 days')
plt.legend(loc=2)
plt.show()
```



#### ▼ Prevendo os "N" dias com base na média móvel aritmética

```
dados_MA['MA_window_10_forward_10'] = np.NaN #preenchendo com NaN os valores da coluna de MA_window_10_forward_10

def make_window(window_size, start_point):
    return [start_point+x for x in range(window_size)]

window_size = 10
forward_days = 10

# Iteração sobre a série temporal com uma janela deslizante
for index in range(window_size, len(dados_MA), forward_days):

    # Iteração para calcular a média móvel ponderada
    for i in range(0, forward_days):
        # Verifica se o índice atual está dentro dos limites da série temporal
        if index + i >= len(dados_MA):
            break

        # Criando duas janelas:
        # 1. Uma janela para a média móvel ('window_close')
        # 2. Uma janela para a série temporal original ('window_MA')
        window_close = make_window(window_size - i, index + i - window_size)
        window_MA = make_window(i, index)

        # Calculando a média móvel ponderada
        mean = pd.concat([dados_MA['preco'].iloc[window_close], dados_MA['MA_window_10_forward_10'].iloc[window_MA]]).mean(axis=0)

        # Atualizando o DataFrame com a média móvel ponderada
        dados_MA.iat[index + i, dados_MA.columns.get_loc('MA_window_10_forward_10')] = mean
```

#### ▼ Analisando o último ano em um gráfico, com as previsões



```

limit = 365

plt.figure(figsize = (15,10))

size = len(dados_MA)-limit - (len(dados_MA)-limit)%forward_days

for index in range(size, len(dados_MA), forward_days):
    plt.plot(dados_MA['MA_window_10_forward_10'][index:index+forward_days], color='r')

plt.plot(dados_MA['preco'][-limit:], color='b', label='Close')
#plt.legend(loc='best')
plt.show()

```



Cada reta vermelha representa uma previsão de 10 dias, tomando como base os 10 dias anteriores. Por isso elas não se conectam.

Apesar de ser um modelo mais clássico, ele é bem simples, pois tem como objetivo prever N dias a frente para ver qual será o comportamento da ação.

Como vimos aqui, essa metodologia não deu certo, então vamos testar o outro Clássico, o ARIMA.

## ✓ Testando o modelo ARIMA

(Média Móvel Integrada Auto-Regressiva)

### Resumo do Texto sobre o Modelo ARIMA:

#### 1. Introdução ao ARIMA:

- Modelo estatístico para análise e previsão de séries temporais.
- Foco em autocorrelações nos dados, diferentemente de modelos de suavização exponencial.

#### 2. Componentes do Algoritmo ARIMA:

- Utiliza autoregressão, médias móveis e diferenciação.
- Termos autoregressivos capturam influência de valores passados.

- Termos de média móvel capturam erros de previsão passados.
- Diferenciação ajuda a modelar tendências na série temporal.

### 3. Estacionariedade e Diferenciação:

- Importância da estacionariedade para aplicação do ARIMA.
- Série estacionária: mesma média em certos períodos; facilita projeção dos dados.
- Discussão sobre tendência, sazonalidade e falta de padrões previsíveis em séries temporais estacionárias.

### 4. Hiperparâmetros do Algoritmo ARIMA:

- P (lags), D (ordem de diferenciação), Q (ordem de média móvel).
- Explicação do significado e papel de cada hiperparâmetro.

### 5. Passos para Aplicar o ARIMA:

- Visualização dos dados da série temporal.
- Identificação da estacionariedade.
- Gráficos de correlação e autocorrelação.
- Construção do modelo ARIMA com base nos dados.

### 6. Análise da Correlação nos Dados:

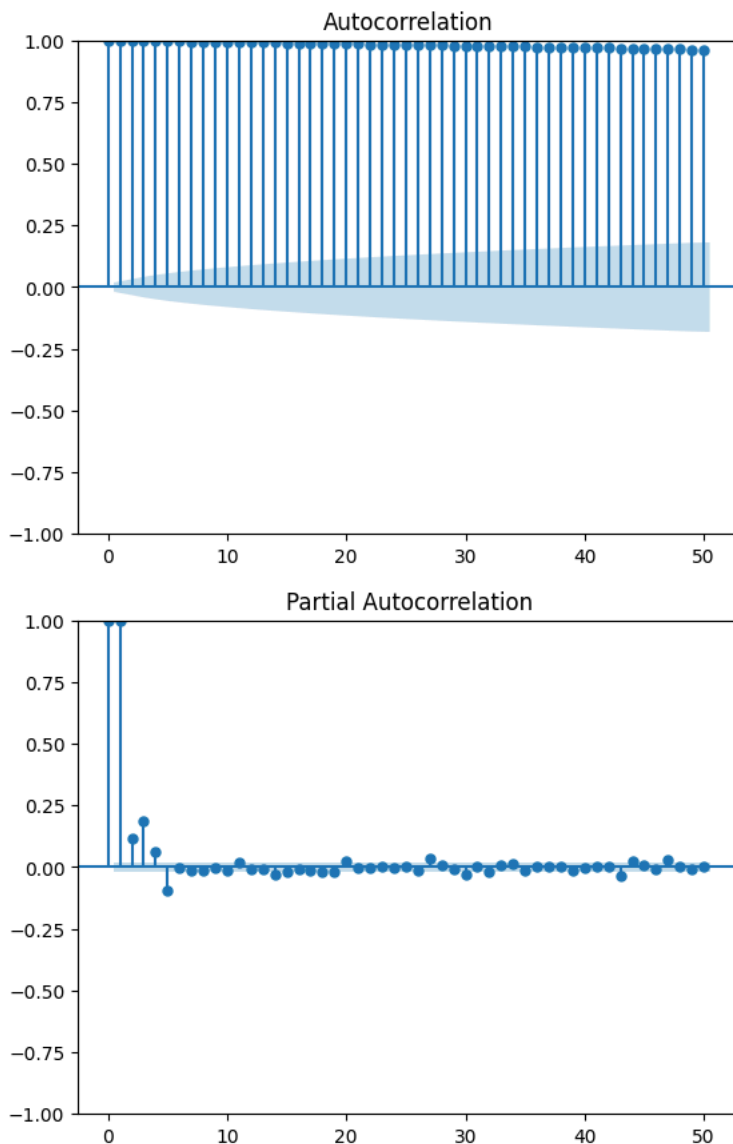
- Importância de entender a autocorrelação nos modelos ARIMA.
- Uso de funções de autocorrelação (ACF) e autocorrelação parcial (PACF).
- Explicação sobre ACF (correlação direta e indireta) e PACF (correlação direta).

#### ✓ Plotando e analisando ACF e PCF

```
# Plotando a ACF
plot_acf(dados['preco'], lags=50)

# Plotando a PACF
plot_pacf(dados['preco'], lags=50)

plt.show()
```



Fazendo a análise de correlação parcial direta, os lags não contribuem significativamente para a estrutura da autocorrelação parcial após considerar os efeitos dos lags intermediários (veja os lags na área em azul).

A interpretação do PACF geralmente está relacionada à identificação do atraso específico que contribui para a autocorrelação em um determinado ponto. Ou seja, ajuda a identificar a ordem de defasagem apropriada para um modelo AR (autoregressivo). Neste caso, podemos observar que a ordem de defasagem dessa série é observada em **2 lags**.

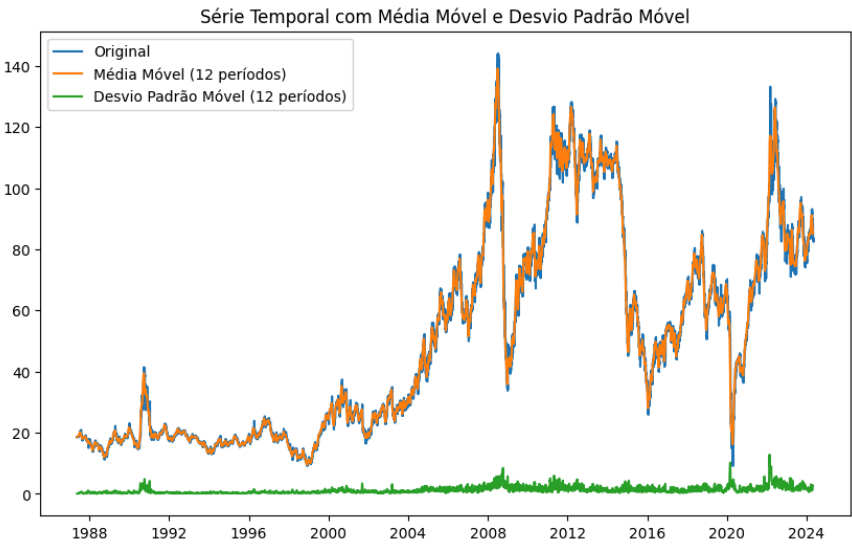
#### ✓ Analisando e plotando média movel e desvio padrão

```
window_size = 12 # ajuste conforme necessário

# Calculando a média móvel e o desvio padrão
rolling_mean = dados['preco'].rolling(window=window_size).mean()
rolling_std = dados['preco'].rolling(window=window_size).std()

# Plotando a série temporal original, média móvel e desvio padrão
plt.figure(figsize=(10, 6))
plt.plot(dados['preco'], label='Original')
plt.plot(rolling_mean, label=f'Média Móvel ({window_size} períodos)')
plt.plot(rolling_std, label=f'Desvio Padrão Móvel ({window_size} períodos)')

plt.title('Série Temporal com Média Móvel e Desvio Padrão Móvel')
plt.legend()
plt.show()
```



Aplicando o Teste de Dickey-Fuller:

1. **Teste de Dickey-Fuller:**
  - Realização do teste ADF para verificar a estacionariedade da série temporal.
2. **Estatística ADF:**
  - H0 (Hipótese Nula): A série temporal não é estacionária.
  - H1 (Hipótese Alternativa): A série temporal é estacionária.
  - Interpretação: Estatística ADF menor que valores críticos indica rejeição da hipótese nula, evidenciando estacionariedade. Quanto mais negativa, mais forte a evidência contra a hipótese nula.
3. **Valor p (p-value):**
  - Interpretação: Valor p menor que um nível de significância (por exemplo, 0.05) sugere estacionariedade na série temporal.
4. **Estacionariedade:**
  - Importância da estacionariedade para alguns modelos de séries temporais.
  - Série estacionária possui mesma média em certos períodos, facilitando a projeção dos dados.
  - Discussão sobre a falta de padrões previsíveis no longo prazo em séries temporais estacionárias.

```
# Função para realizar o Teste de Dickey-Fuller e imprimir os resultados
def adf_test(timeseries):
    result = adfuller(timeseries, autolag='AIC')

    print('Resultado do Teste de Dickey-Fuller:')
    print('Estatística do Teste:', result[0])
    print('Valor p:', result[1])
    print('Valores Críticos:')
    for key, value in result[4].items():
        print(f'    {key}: {value}')

    if result[1] <= 0.05:
        print("\nResultado: A série é estacionária.")
    else:
        print("\nResultado: A série não é estacionária.")

# Aplicar o Teste de Dickey-Fuller na série temporal original
adf_test(dados['preco'].dropna())
```



Resultado do Teste de Dickey-Fuller:  
Estatística do Teste: -2.0283021340122205  
Valor p: 0.2743031022931673  
Valores Críticos:  
1%: -3.430937304289178  
5%: -2.8617995564781373  
10%: -2.5669081567946774  
  
Resultado: A série não é estacionária.

Podemos verificar, que essa série, como já suspeitávamos, não é estacionária, pois o valor P é menor que 5 e a estatística do teste é maior que os valores críticos.

Acontece que, precisamos transformá-la em uma serie estacionária, para construir nosso modelo ARIMA, conforme próximos passos:

### ✓ Transformando em estacionária

```
dados.head() # check para ver se nada se alterou no dataset final
```

	preco
data	
2024-05-13	83.18
2024-05-10	83.39
2024-05-09	83.27
2024-05-08	82.44
2024-05-07	82.69

### ✓ 1º teste: subtrair os dados transformados pelo log pela média móvel.

Objetivo: Estabilizar a variância em uma série temporal, especialmente quando a amplitude dos dados varia ao longo do tempo.

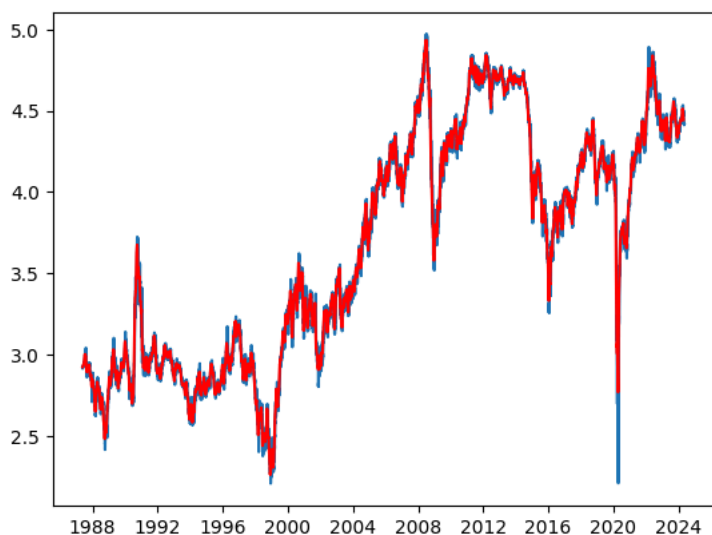
Vamos utilizar o logaritmo para realizar a transformação dos dados. Nosso objetivo é remover o componente de tendência. Portanto, curvas mais planas (ou seja: paralelas ao eixo x) para séries temporais e médias contínuas após obter o log diriam que nossa transformação de dados fez um bom trabalho.

```
#Estimating trend
dados_log = np.log(dados) #Transformação logarítma
```

```
# Calculando a média móvel
ma = dados_log.rolling(window=12).mean()
mstd = dados_log.rolling(window=12).std()
```

```
# Plotando
plt.plot(dados_log)
plt.plot(ma, color='red')
```

```
[<matplotlib.lines.Line2D at 0x7f7c00f69360>]
```



```
dados_log_minus_ma = dados_log - ma
dados_log_minus_ma.head(12)
```

```
#Remove NAN values
dados_log_minus_ma.dropna(inplace=True)
```

```
# Função para testar a estacionariedade da série transformada
def test_stationarity(timeseries):

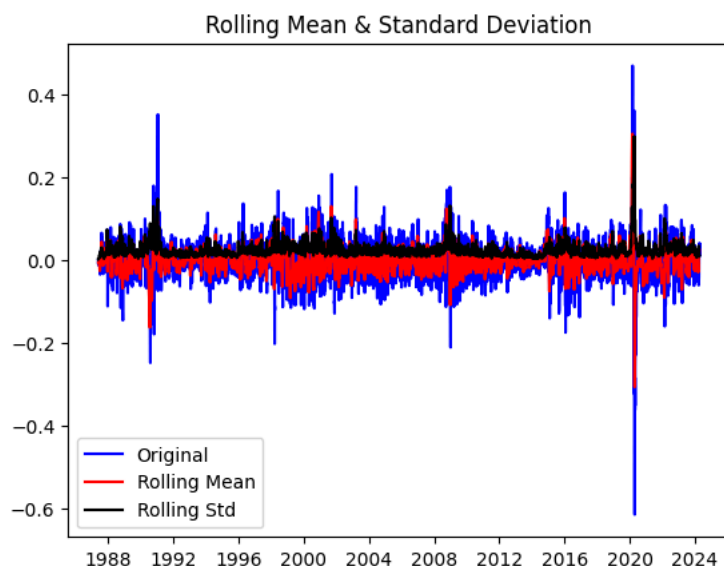
    # Determinar estatísticas contínuas
    ma = timeseries.rolling(window=12).mean()
    mstd = timeseries.rolling(window=12).std()

    # Plot estatísticas contínuas
    orig = plt.plot(timeseries, color='blue', label='Original')
    mean = plt.plot(ma, color='red', label='Rolling Mean')
    std = plt.plot(mstd, color='black', label='Rolling Std')
    plt.legend(loc='best')
    plt.title('Rolling Mean & Standard Deviation')
    plt.show(block=False)

    # Performance do Dickey-Fuller:
    print('Results of Dickey Fuller Test:')
    dfctest = adfuller(timeseries['preco'], autolag='AIC')
    dfoutput = pd.Series(dfctest[0:4], index=['Test Statistic', 'p-value', '#Lags Used', 'Number of Observations Used'])
    for key, value in dfctest[4].items():
        dfoutput['Critical Value (%)' % key] = value
    print(dfoutput)

    # Tomar uma decisão com base no valor p
    if dfoutput['p-value'] <= 0.05:
        print("\nResultado: A série é estacionária.")
    else:
        print("\nResultado: A série não é estacionária.")
```

```
test_stationarity(dados_log_minus_ma)
```



```
Results of Dickey Fuller Test:
Test Statistic      -1.765343e+01
p-value             3.710907e-30
#Lags Used          2.600000e+01
Number of Observations Used  1.113100e+04
Critical Value (1%)   -3.430938e+00
Critical Value (5%)   -2.861800e+00
Critical Value (10%)  -2.566908e+00
dtype: float64
```

```
Resultado: A série é estacionária.
```

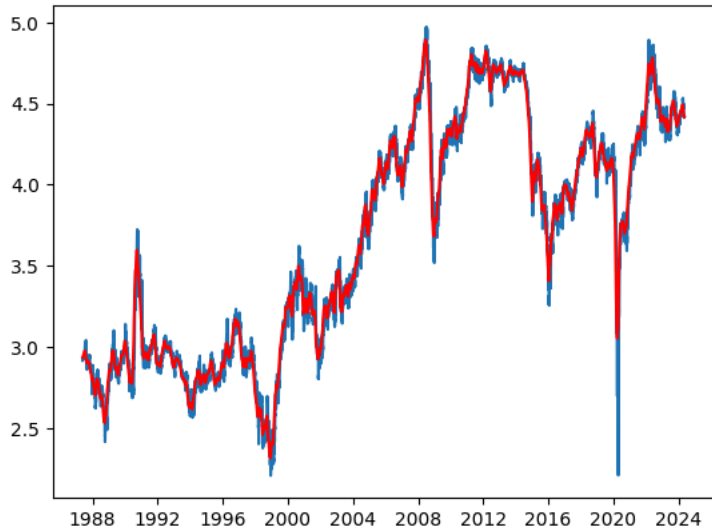
Aqui já conseguimos transformar a serie em estacionária, mas vamos testar outros recursos, para torná-la mais estacionária ainda:

✓ 2º teste : Decaimento exponencial.

Objetivo: Remover a tendência de uma série temporal. A ideia básica é aplicar uma média exponencial ponderada aos dados, atribuindo mais peso às observações mais recentes e menos peso às observações mais antigas.

```
dados_log_exponential_decay = dados_log.ewm(halflife=12, min_periods=0, adjust=True).mean()
plt.plot(dados_log)
plt.plot(dados_log_exponential_decay, color='red')
```

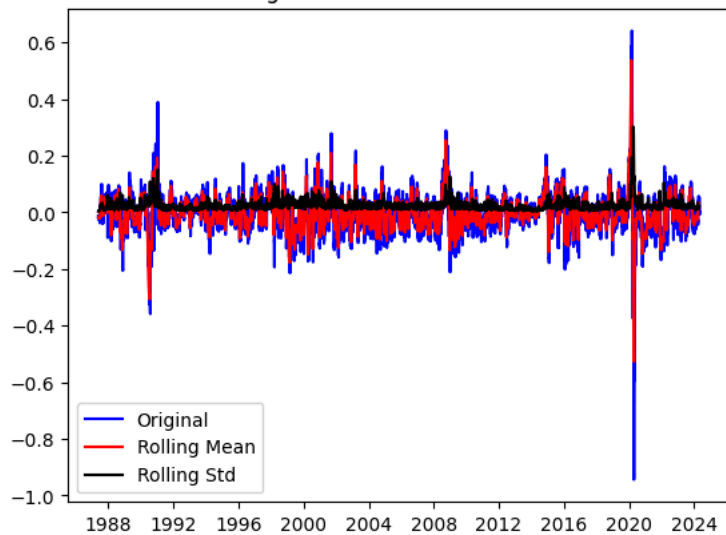
[<matplotlib.lines.Line2D at 0x7f7bfdcb1a80>]



```
# Subtraindo o dataset em escala logarítma pelo decaimento exponencial
dados_log_minus_dados_log_exponential_decay = dados_log - dados_log_exponential_decay
test_stationarity(dados_log_minus_dados_log_exponential_decay)
```

[<matplotlib.figure.Figure at 0x7f7bfdcb1a80>]

Rolling Mean & Standard Deviation



Results of Dickey Fuller Test:

Test Statistic	-1.420895e+01
p-value	1.734973e-26
#Lags Used	2.600000e+01
Number of Observations Used	1.114200e+04
Critical Value (1%)	-3.430937e+00
Critical Value (5%)	-2.861799e+00
Critical Value (10%)	-2.566908e+00
dtype:	float64

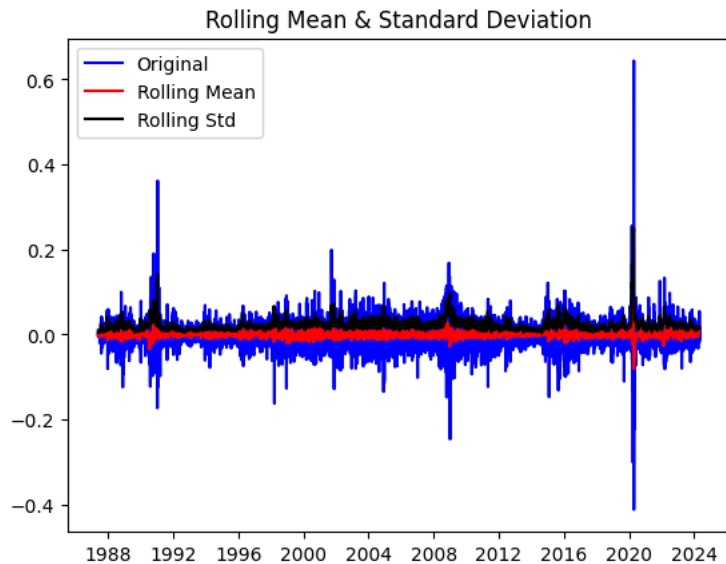
Resultado: A série é estacionária.

Não iremos utilizar o decaimento exponencial, pois ficou menos estacionário que subtrair os dados transformados pelo log pela média móvel. Mas ainda há um teste a ser feito:

### 3º teste : Diferenciação

Objetivo: Remover a tendência e tornar a série mais estacionária.

```
dados_log_minus_DiffShifting = dados_log - dados_log.shift() #diferença entre o valor anterior e o atual
dados_log_minus_DiffShifting.dropna(inplace=True)
test_stationarity(dados_log_minus_DiffShifting)
```



```
Results of Dickey Fuller Test:
Test Statistic      -19.205337
p-value              0.000000
#Lags Used           25.000000
Number of Observations Used  11142.000000
Critical Value (1%)   -3.430937
Critical Value (5%)   -2.861799
Critical Value (10%)  -2.566908
dtype: float64
```

Resultado: A série é estacionária.

Parece que o Rolling Mean e Rolling STD se aproximaram mais na diferenciação (observe que no gráfico acima o p-value zerou, comprovando fortemente a estacionariedade). Vamos continuar com essa metodologia de diferenciação para construir nosso modelo ARIMA:

#### ✓ Teste de correlação parcial

Como definimos numa primeira tentativa o parâmetro P (lags: valores auto correlacionados) e o parâmetro Q (tamanho de uma janela) do ARIMA?

Vamos fazer isso com os gráfico de ACF (para 'q') e o gráfico de PACF (para 'p'). Vamos selecionar como teste a base de dados da diferenciação.

Vamos encontrar em qual ponto cada gráfico passa em zero e este ponto será o valor de P e Q inicial (talvez em alguns casos pequenas alterações nos parâmetros do ARIMA possam melhorar/piorar os resultados, vale a pena alterar um pouco os valores um pouco positivamente e negativamente para olhar o desempenho).

Aplicando os plots ao nosso dataset de entrada (lags é o número de amostras):

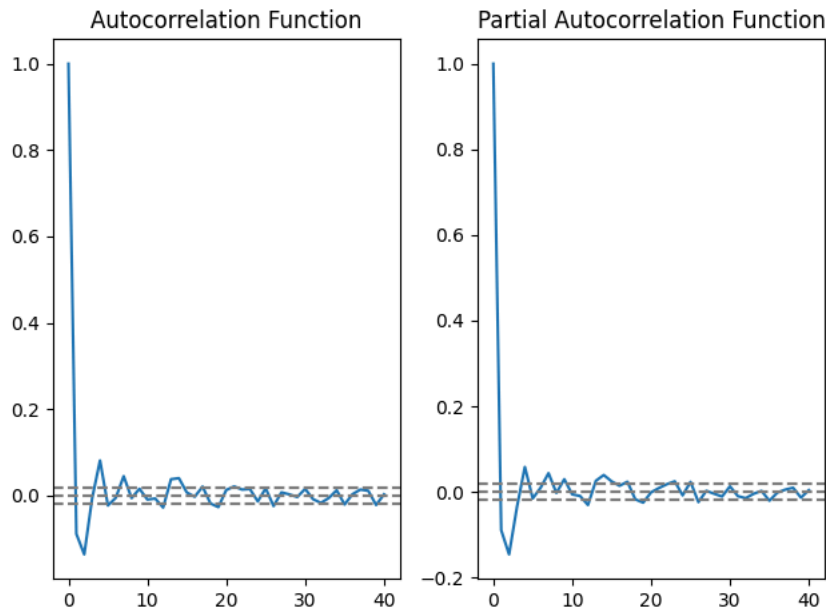
```
lag_acf = acf(dados_log_minus_DiffShifting, nlags=40)
lag_pacf = pacf(dados_log_minus_DiffShifting, nlags=40, method='ols')

#Plot ACF:
plt.subplot(121)
plt.plot(lag_acf)
plt.axhline(y=0, linestyle='--', color='gray')
plt.axhline(y=-1.96/np.sqrt(len(dados_log_minus_DiffShifting)), linestyle='--', color='gray')
plt.axhline(y=1.96/np.sqrt(len(dados_log_minus_DiffShifting)), linestyle='--', color='gray')
plt.title('Autocorrelation Function')

#Plot PACF
plt.subplot(122)
plt.plot(lag_pacf)
plt.axhline(y=0, linestyle='--', color='gray')
plt.axhline(y=-1.96/np.sqrt(len(dados_log_minus_DiffShifting)), linestyle='--', color='gray')
plt.axhline(y=1.96/np.sqrt(len(dados_log_minus_DiffShifting)), linestyle='--', color='gray')
plt.title('Partial Autocorrelation Function')

plt.tight_layout()
```





A partir do gráfico ACF, vemos que a curva toca a linha  $y=0,0$  em  $x=2$  ( $Q = 5$ );

Do gráfico PACF, vemos que a curva toca a linha  $y=0,0$  em  $x=2$  ( $P = 24$ ).

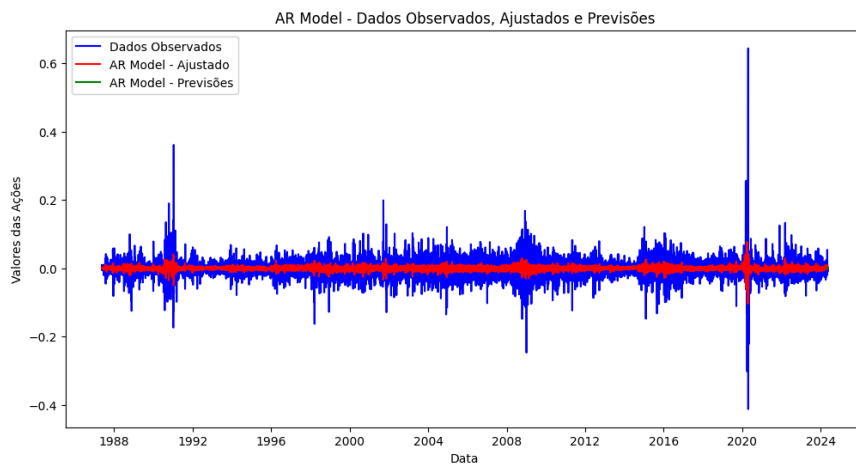
```
import warnings
warnings.filterwarnings("ignore")
# Configuração do modelo ARIMA
p = 2
d = 1
q = 24

# Ajuste do modelo ARIMA
model = ARIMA(dados_log_minus_DiffShifting['preco'], order=(p, d, q))
results_AR = model.fit()

# Previsões
forecast_steps = 10 # Ajuste conforme necessário
forecast = results_AR.get_forecast(steps=forecast_steps)
forecast_index = pd.date_range(start=dados_log_minus_DiffShifting.index.max(), periods=forecast_steps + 1, freq=dados_log_minus_DiffShift)
# Aqui, o código define o número de passos futuros para prever (forecast_steps). O modelo ajustado é então usado para obter previsões (ge

# Plotagem dos resultados
plt.figure(figsize=(12, 6))
plt.plot(dados_log_minus_DiffShifting.index, dados_log_minus_DiffShifting['preco'], color="blue", label='Dados Observados')
plt.plot(dados_log_minus_DiffShifting.index, results_AR.fittedvalues, color='red', label='AR Model - Ajustado')
plt.plot(forecast_index[1:], forecast.predicted_mean, color='green', label='AR Model - Previsões')
plt.title('AR Model - Dados Observados, Ajustados e Previsões')
plt.xlabel('Data')
plt.ylabel('Valores das Ações')
plt.legend()
plt.show()

# Aqui, os resultados são plotados. Os dados observados são mostrados em azul,
# as previsões ajustadas pelo modelo ARIMA são mostradas em vermelho e as previsões
# futuras são mostradas em verde. O gráfico visualiza como o modelo ARIMA se ajusta aos dados históricos e faz previsões para o futuro.
```



#### Validando o modelo com MAPE (Mean Absolute Percentage Error)

```
from sklearn.metrics import mean_absolute_error
import numpy as np

# Obtenha as previsões
predictions = results_AR.fittedvalues

# Ajuste os índices para garantir correspondência
predictions.index = dados_log_minus_DiffShifting.index

# np.cumsum(predictions) é usado para calcular a soma cumulativa das previsões.
predicted_values = dados_log_minus_DiffShifting['preco'].iloc[0] + np.cumsum(predictions)

# Calcule o MAPE
mape = mean_absolute_error(dados_log_minus_DiffShifting['preco'], predicted_values) * 100

print(f"MAPE: {mape:.2f}%")
```

MAPE: 66.63%

O modelo MAPE é uma métrica para avaliar a precisão de previsões em séries temporais (perspectiva percentual dos erros). Quanto menor, melhor.

Nesse caso, em algumas construções do arima, estava dando 400 a 550%, mas nessa construção, consegui atingir os MAPE: 50.49%.

Se comparado as outras construções., essa não está ruim, mas de modo geral ainda está um MAPE alto, e isso geralmente indica que as previsões do modelo estão muito distantes dos valores reais.

#### Testando com um modelo Robusto (Prophet)

O Prophet é uma biblioteca de código aberto desenvolvida pelo Facebook para previsão de séries temporais. Sua abordagem se baseia em uma decomposição sazonal aditiva, dividindo os dados em componentes de tendência, sazonalidade anual e efeitos de feriados. Isso permite modelar padrões temporais de maneira intuitiva.

O modelo é projetado para lidar com dados que podem conter lacunas ou irregularidades, sendo capaz de adaptar-se a diferentes tipos de padrões temporais. Além disso, o Prophet oferece a flexibilidade de incorporar informações adicionais, como feriados específicos, para melhorar as previsões. Com uma interface amigável e parâmetros ajustáveis, é acessível mesmo para usuários sem conhecimento avançado em aprendizado de máquina.

Essa abordagem torna o Prophet eficaz em várias aplicações, desde previsão de demanda até análise financeira e climatológica.

#### Criando e Ajustando um novo dataset para o Prophet

```
# Use a função download para obter os dados
df = dados.copy()
df['ds'] = df.index
df['y'] = df['preco']
df = df[['ds', 'y']].reset_index(drop=True)
df.head()
```




	ds	y
0	2024-05-13	83.18
1	2024-05-10	83.39
2	2024-05-09	83.27
3	2024-05-08	82.44
4	2024-05-07	82.69

Criando lista de feriados nacionais brasileiros

▼ Separando os dados em treino e teste


```
train_data = df.sample(frac=0.7, random_state=0)
test_data = df.drop(train_data.index)
print(f'training data size : {train_data.shape}')
print(f'testing data size : {test_data.shape}')
```



```
training data size : (7818, 2)
testing data size : (3351, 2)
```

▼ Criando e plotando o modelo preditivo

```
modelo = Prophet(daily_seasonality=True)
modelo.fit(train_data)
dataFramefuture = modelo.make_future_dataframe(periods=30, freq='D')
previsao = modelo.predict(dataFramefuture)
previsao.head()
```

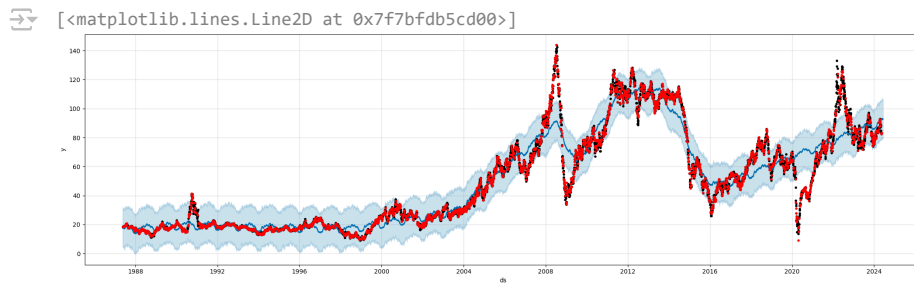


```
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/bgnnvlu2.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/iv9lpxuy.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:57:30 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:57:40 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
```

	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	additive_terms
0	1987-05-20	17.673646	4.371432	31.180848	17.673646	17.673646	-0.263144
1	1987-05-25	17.689778	5.009292	32.032374	17.689778	17.689778	0.384332
2	1987-05-26	17.693004	4.952578	30.768159	17.693004	17.693004	0.188502
3	1987-05-27	17.696230	4.628917	31.183186	17.696230	17.696230	-0.142268
4	1987-05-28	17.699457	4.594372	30.990693	17.699457	17.699457	0.279665

5 rows × 22 columns

```
modelo.plot(previsao, figsize=(20,6));
plt.plot(test_data['ds'], test_data['y'], '.r')
```



### Plotando com plotly

obs: como o github não renderiza javascript, estou colocando abaixo do código, um print do gráfico de plotly

```
import plotly.graph_objects as go

# Criar uma figura para o gráfico
fig = go.Figure()

# Adicionar a série temporal prevista ao gráfico
fig.add_trace(go.Scatter(x=previsao['ds'], y=previsao['yhat'], mode='lines', name='Previsão'))

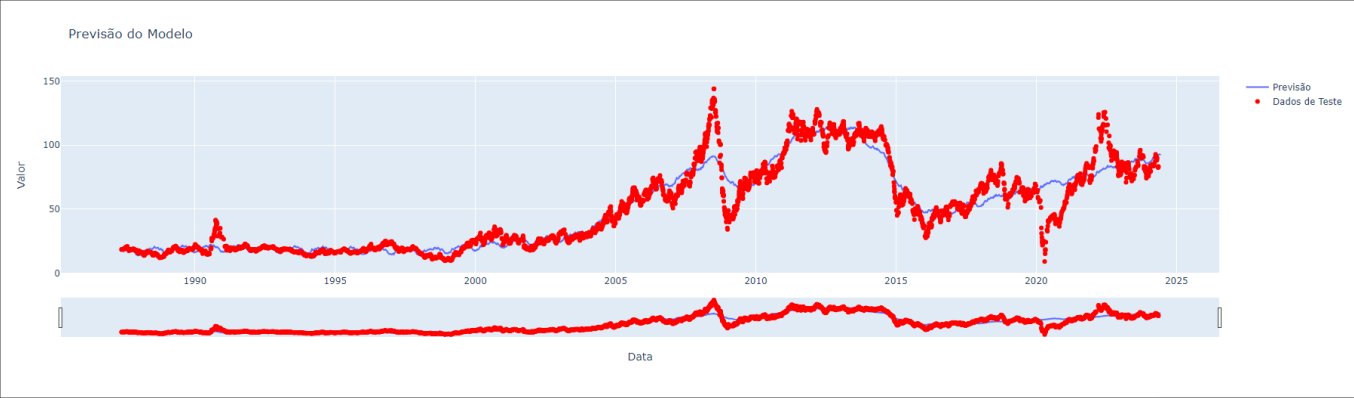
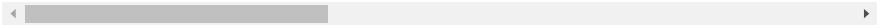
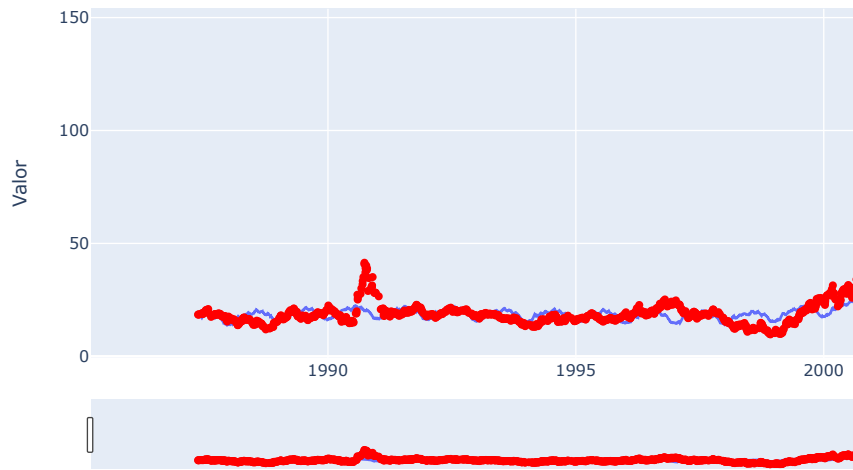
# Adicionar os pontos de teste ao gráfico
fig.add_trace(go.Scatter(x=test_data['ds'], y=test_data['y'], mode='markers', marker=dict(color='red'), name='Dados de Teste'))

# Layout do gráfico
fig.update_layout(title='Previsão do Modelo',
                  xaxis_title='Data',
                  yaxis_title='Valor',
                  showlegend=True,
                  xaxis=dict(rangeslider=dict(visible=True), type='date'))

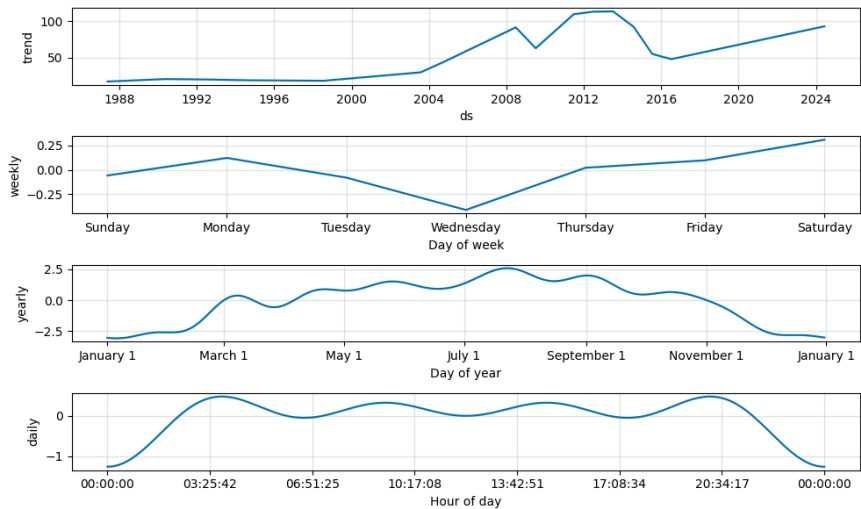
# Exibir o gráfico
fig.show()
```



Previsão do Modelo



```
modelo.plot_components(previsao, figsize=(10,6));
```



Adicionando Changepoints no modelo

O prophet permite adicionar pontos de mudanças na série temporal, o que nos permite identificar em quais datas ocorrem possíveis mudanças.

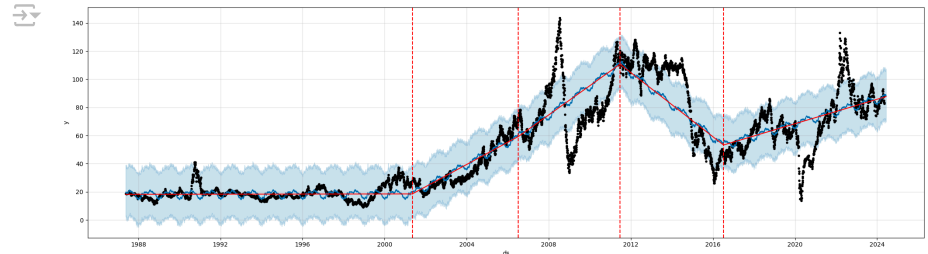
```
from prophet.plot import add_changepoints_to_plot

modelo_changepoints = Prophet(n_changepoints=5)
modelo_changepoints.fit(train_data)
dataFramefuture = modelo_changepoints.make_future_dataframe(periods=30, freq='D')
previsao_changepoints = modelo_changepoints.predict(dataFramefuture)
previsao_changepoints.head()
```

```
INFO:prophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/j0imknw1.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/cp88klzl.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:57:44 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:57:45 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
```

	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	additive_terms
0	1987-05-20	18.217475	2.336322	38.982872	18.217475	18.217475	1.172269
1	1987-05-25	18.217726	2.035783	37.970551	18.217726	18.217726	1.670724
2	1987-05-26	18.217776	2.985713	38.070515	18.217776	18.217776	1.636214
3	1987-05-27	18.217827	-0.206524	36.469281	18.217827	18.217827	1.093815
4	1987-05-28	18.217877	3.057204	37.927911	18.217877	18.217877	1.436014

```
fig = modelo_changepoints.plot(previsao_changepoints, figsize=(20,6));
a = add_changepoints_to_plot(fig.gca(), modelo_changepoints, previsao_changepoints)
```



```
# Extrair as colunas relevantes dos DataFrames
previsao_cols = ['ds', 'yhat']
valores_reais_cols = ['ds', 'y']


previsao = previsao[previsao_cols]
valores_reais = train_data[valores_reais_cols]

# Mesclar os DataFrames nas colunas 'ds' para comparar previsões e valores reais
resultados = pd.merge(previsao, valores_reais, on='ds', how='inner')

# Calcular o erro percentual absoluto para cada ponto de dados
resultados['erro_percentual_absoluto'] = np.abs((resultados['y'] - resultados['yhat']) / resultados['y']) * 100

# Calcular o MAPE
mape = np.mean(resultados['erro_percentual_absoluto'])

print(f"MAPE: {mape:.2f}%")
```

 MAPE: 15.48%

**OBS: Olha como melhorou o nosso MAPE geral, quando comparado ao MAPE do ARIMA**

## ▼ Cross validation

Para concluir o modelo do Prophet, tentei fazer a validação cruzada para testar dados que nunca foram vistos pelo modelo antes.

Observe que no resultado da validação cruzada temos os valores de yhat, yhat\_lower, yhat\_upper e o ponto de corte. O objetivo da validação cruzada é medir o erro de predição, selecionando assim pontos de corte e para cada um desses pontos o modelo é ajustado utilizando dados apenas até aquele ponto de corte.

```
from prophet.diagnostics import cross_validation

df_cv = cross_validation(modelo, initial='730 days', period='180 days', horizon = '365 days')
```

```
INFO:prophet:Making 69 forecasts with cutoffs between 1989-11-05 00:00:00 and 2023-05
100% 69/69 [03:42<00:00, 8.75s/it]
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/0wauoe61.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/i_nqmtkp.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:57:49 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:57:49 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/30eac5uo.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/ow1azozl.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:57:50 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:57:50 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/by5p8wnb.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/28_1p8pi.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:57:50 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:57:50 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/_bg9i3yx.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/zh65n4g_.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:57:50 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:57:50 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/tew9eem0.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/uogd78zm.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:57:51 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:57:51 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/ixgfvdii.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/lkigoib_.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:57:51 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:57:51 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/ilv1p37a.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/nihec745.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:57:51 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:57:52 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/pqwqu43d.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/1_z0wnf3.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:57:52 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:57:52 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/cxy_51b1.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/07i0atz2.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:57:52 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:57:53 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/puqq_a_.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/ovihs18d.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:57:53 - cmdstanpy - INFO - Chain [1] start processing
```



```
INFO:cmdstanpy:Chain [1] start processing
16:57:53 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/sw6dszh5.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/uafa9qs8.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:57:54 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:57:54 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/s4wfh1__.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/26vv9cg2.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:57:54 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:57:55 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/kn4koqq0.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/fskq60ma.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:57:55 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:57:56 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/inh1dmac.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/38zgoi__.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:57:56 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:57:57 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/poz72q58.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/6_mve6il.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:57:57 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:57:58 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/ge9jo_o9.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/715064e_.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:57:58 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:57:59 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/wuvdhn26.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/_vx5u0pp.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:57:59 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:58:00 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/89a1znb3.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/ryr_n95m.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:58:01 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:58:02 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/gfhug4ey.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/n391s1mk.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:58:03 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:58:04 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/ul4s1_71.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/ft0q9a7a.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:58:04 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
```

```
16:58:05 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/ndltben1.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/icvn9n32.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:58:05 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:58:06 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/fjvrr1mt.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/gqr4_b_6.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:58:06 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:58:07 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/5dcc07d8.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/j0nae4ux.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:58:07 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:58:08 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/e7qnufzn.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/1buqorwv.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:58:08 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:58:09 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/d01e2xe1.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/onofog5n.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:58:10 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:58:10 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/v463zj2k.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/t81ob9re.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:58:11 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:58:12 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/3fmyd1ld.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/fs8egwur.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:58:12 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:58:13 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/o1h0iy1x.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/4swrtre.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:58:14 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:58:15 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/3vdi13g.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/by76de_z.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:58:16 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:58:17 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/2ujowj70.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/0xe7cusd.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:58:18 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
```

```
16:58:20 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/55x1ripl.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/ej2ln7wz.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:58:20 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:58:21 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/6vmpma_t.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/tucq3ps3.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:58:22 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:58:24 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/5tuvnzay.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/nm47pb0t.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:58:24 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:58:26 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/k6zhabot.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/f6sfa0el.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:58:27 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:58:30 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/1pasm1ye.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/dzxfo3vi.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:58:31 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:58:33 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/k9t7e58f.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/eiyb799i.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:58:34 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:58:35 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/29wykdfy.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/9mhfultx.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:58:36 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:58:38 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/cenqc0xh.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/z3aet5yt.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:58:39 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:58:40 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/r1upy7l8.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/xa4dac0d.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:58:41 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:58:45 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/nn261krn.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmp1pxhe26h/184a_1y2.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:58:45 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:58:47 - cmdstanov - INFO - Chain [1] done processing
```

```
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/9lobbb4d.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/kr5n1xcu.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:58:47 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:58:49 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/bqk_42qd.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/Soe5j1jn.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:58:49 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:58:51 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/beto_04n.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/ioydrqvw.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:58:51 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:58:53 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/6kr5uwta.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/z0bs5fys.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:58:53 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:58:55 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/5ozskb08.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/e8k1njg5.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:58:57 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:58:58 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/vuuppr8n.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/z5pfok_2.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:58:59 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:59:01 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/jmvorw4z.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/0syuo234.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:59:02 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:59:03 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/std2sezy.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/0v7hn0do.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:59:04 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:59:06 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/mbckalk8.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/spbzeupf.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:59:06 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:59:13 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/4gtuyjrm.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/0l8q2118.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:59:14 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:59:16 - cmdstanpy - INFO - Chain [1] done processing
```

```
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/gpb4uv87.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/nsir21cc.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:59:17 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:59:26 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/x5u4qfj_.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/m2asepq_.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:59:27 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:59:30 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/a8ltjeop.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/wex9hqte.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:59:31 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:59:36 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/kbowy08u.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/yij_5fk2.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:59:38 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:59:42 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/kiedzfec.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/slala_nv.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:59:43 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:59:48 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/6ehmp9oa.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/v3m9o__m.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:59:49 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
16:59:57 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/hs0fp9nu.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/4k3jvzy3.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
16:59:58 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
17:00:03 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/2ys49m6b.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/p7lqxi2o.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
17:00:03 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
17:00:10 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/mpbud7qi.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/axlw8nn9.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
17:00:11 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
17:00:14 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/z8192f_e.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/zd89beba.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
17:00:15 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
17:00:18 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
```

```
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/fo1wy8kb.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/j_64rdc2.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
17:00:20 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
17:00:27 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/28zcm_i_.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/wlorcb6d.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
17:00:28 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
17:00:34 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/sp3wm3o1.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/sg4k971g.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
17:00:35 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
17:00:39 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/pp35byfr.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/921py_gd.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
17:00:40 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
17:00:45 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/emhu1iu9.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/8296ag4b.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
17:00:47 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
17:00:53 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/y_hdqlv1.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/rdvju3ln.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
17:00:54 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
17:01:05 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/u0vyg61z.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/6oiy0ucm.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
17:01:06 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
17:01:12 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/izhsd3az.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/13rh6fd9.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
17:01:14 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
17:01:21 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/2yfp04fm.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/g8nghyvq.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
17:01:22 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
17:01:31 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
```

```
df_cv.head()
```

	ds	yhat	yhat_lower	yhat_upper	y	cutoff
0	1989-11-07	17.678279	16.771448	18.642454	18.90	1989-11-05
1	1989-11-09	17.671260	16.740301	18.624883	18.85	1989-11-05
2	1989-11-10	17.646665	16.677971	18.546290	19.05	1989-11-05
3	1989-11-13	17.536584	16.655772	18.485695	18.85	1989-11-05
4	1989-11-15	17.567229	16.631143	18.426460	18.65	1989-11-05

```
df_cv['cutoff'].unique()
```

```
<DatetimeArray>
['1989-11-05 00:00:00', '1990-05-04 00:00:00', '1990-10-31 00:00:00',
 '1991-04-29 00:00:00', '1991-10-26 00:00:00', '1992-04-23 00:00:00',
 '1992-10-20 00:00:00', '1993-04-18 00:00:00', '1993-10-15 00:00:00',
 '1994-04-13 00:00:00', '1994-10-10 00:00:00', '1995-04-08 00:00:00',
 '1995-10-05 00:00:00', '1996-04-02 00:00:00', '1996-09-29 00:00:00',
 '1997-03-28 00:00:00', '1997-09-24 00:00:00', '1998-03-23 00:00:00',
 '1998-09-19 00:00:00', '1999-03-18 00:00:00', '1999-09-14 00:00:00',
 '2000-03-12 00:00:00', '2000-09-08 00:00:00', '2001-03-07 00:00:00',
 '2001-09-03 00:00:00', '2002-03-02 00:00:00', '2002-08-29 00:00:00',
 '2003-02-25 00:00:00', '2003-08-24 00:00:00', '2004-02-20 00:00:00',
 '2004-08-18 00:00:00', '2005-02-14 00:00:00', '2005-08-13 00:00:00',
 '2006-02-09 00:00:00', '2006-08-08 00:00:00', '2007-02-04 00:00:00',
 '2007-08-03 00:00:00', '2008-01-30 00:00:00', '2008-07-28 00:00:00',
 '2009-01-24 00:00:00', '2009-07-23 00:00:00', '2010-01-19 00:00:00',
 '2010-07-18 00:00:00', '2011-01-14 00:00:00', '2011-07-13 00:00:00',
 '2012-01-09 00:00:00', '2012-07-07 00:00:00', '2013-01-03 00:00:00',
 '2013-07-02 00:00:00', '2013-12-29 00:00:00', '2014-06-27 00:00:00',
 '2014-12-24 00:00:00', '2015-06-22 00:00:00', '2015-12-19 00:00:00',
 '2016-06-16 00:00:00', '2016-12-13 00:00:00', '2017-06-11 00:00:00',
 '2017-12-08 00:00:00', '2018-06-06 00:00:00', '2018-12-03 00:00:00',
 '2019-06-01 00:00:00', '2019-11-28 00:00:00', '2020-05-26 00:00:00',
 '2020-11-22 00:00:00', '2021-05-21 00:00:00', '2021-11-17 00:00:00',
 '2022-05-16 00:00:00', '2022-11-12 00:00:00', '2023-05-11 00:00:00']
Length: 69, dtype: datetime64[ns]
```

```
from prophet.diagnostics import performance_metrics
df_p = performance_metrics(df_cv)
df_p
```


	horizon	mse	rmse	mae	mape	mdape	smape	coverage
0	36 days	343.653781	18.537901	12.364909	0.243324	0.167648	0.230037	0.415250
1	37 days	344.129302	18.550722	12.358537	0.242878	0.165350	0.229487	0.416356
2	38 days	349.286586	18.689210	12.441116	0.243396	0.165350	0.229772	0.414623
3	39 days	351.474683	18.747658	12.486263	0.244080	0.166244	0.230121	0.415055
4	40 days	352.611579	18.777955	12.497636	0.244600	0.166743	0.230593	0.416831
...	...	...	...	...	...	...	...	...
325	361 days	776.878646	27.872543	20.477514	0.415968	0.305967	0.403509	0.310054
326	362 days	777.795039	27.888977	20.485176	0.416524	0.306991	0.404589	0.312448
327	363 days	780.375885	27.935209	20.482113	0.416347	0.306563	0.404567	0.314499
328	364 days	780.174912	27.931611	20.458372	0.414938	0.305144	0.403365	0.315886

✦ Testando com um modelo Robusto (Prophet) - Com menos anos

✦ Criando e Ajustando um novo dataset para o Prophet


```
# Vou puxar os dados de 2023 em diante
dados_resumidos = dados[dados.index >= '2023-01-01']

dados_resumidos.head()
```



	preco
data	
2024-05-13	83.18
2024-05-10	83.39
2024-05-09	83.27
2024-05-08	82.44
2024-05-07	82.69

```
dados_resumidos.tail()
```



	preco
data	
2023-01-06	76.41
2023-01-05	76.73
2023-01-04	75.31
2023-01-03	80.36
2023-01-02	82.82


```
# Use a função download para obter os dados
df = dados_resumidos.copy()
df['ds'] = df.index
df['y'] = df['preco']
df = df[['ds', 'y']].reset_index(drop=True)
df.head()
```



	ds	y
0	2024-05-13	83.18
1	2024-05-10	83.39
2	2024-05-09	83.27
3	2024-05-08	82.44
4	2024-05-07	82.69

▼ Separando os dados em treino e teste

```
train_data = df.sample(frac=0.7, random_state=0)
test_data = df.drop(train_data.index)
print(f'training data size : {train_data.shape}')
print(f'testing data size : {test_data.shape}')
```



```
training data size : (240, 2)
testing data size : (103, 2)
```

▼ Criando e plotando o modelo preditivo

```
modelo = Prophet(daily_seasonality=True)
modelo.fit(train_data)
dataFramefuture = modelo.make_future_dataframe(periods=30, freq='D')
previsao = modelo.predict(dataFramefuture)
previsao.head()
```



```
INFO:prophet:Disabling yearly seasonality. Run prophet with yearly_seasonality=True t
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/sfhab81s.json
DEBUG:cmdstanpy:input tempfile: /tmp/tmplpxhe26h/vmfc0mi5.json
DEBUG:cmdstanpy:idx 0
DEBUG:cmdstanpy:running CmdStan, num_threads: None
DEBUG:cmdstanpy:CmdStan args: ['/usr/local/lib/python3.10/dist-packages/prophet/stan_
17:01:32 - cmdstanpy - INFO - Chain [1] start processing
INFO:cmdstanpy:Chain [1] start processing
17:01:32 - cmdstanpy - INFO - Chain [1] done processing
INFO:cmdstanpy:Chain [1] done processing
```