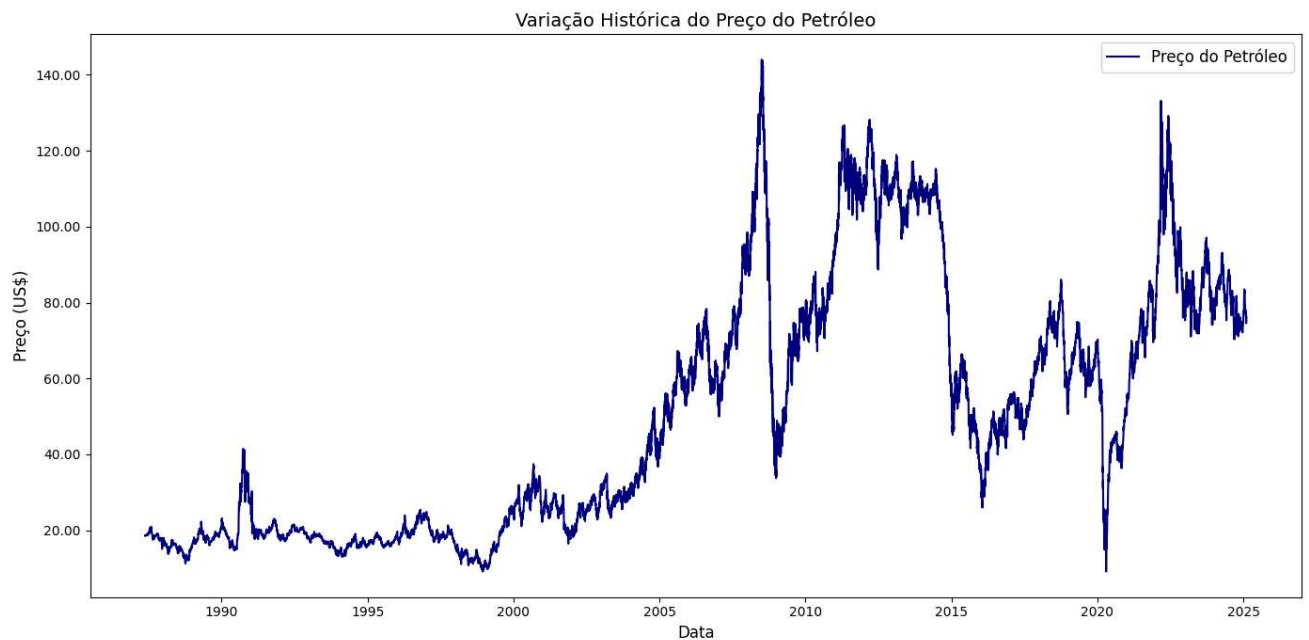


✓ 1. Carregamento e Preparação dos Dados

- Leitura do arquivo CSV e conversão da coluna de datas para o formato datetime.
- Ordenação dos dados pela data para manter a sequência temporal.
- Visualização inicial para entender a variação histórica dos preços.

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from matplotlib.ticker import FuncFormatter
5 from sklearn.preprocessing import MinMaxScaler
6 from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
7 import math
8 import datetime
9
10 # 1. Carregamento e Preparação dos Dados
11 df = pd.read_csv("base_preco_petroleo.csv")
12
13 # Conversão da coluna 'Data' para datetime
14 df['Data'] = pd.to_datetime(df['Data'], dayfirst=True)
15
16 # Ordenação dos dados pela data
17 df.sort_values('Data', inplace=True)
18 df.reset_index(drop=True, inplace=True)
19
20 # Conversão dos preços: remove espaços e substitui vírgula por ponto, convertendo para
21 df['Preço do Petroleo'] = df['Preço do Petroleo'].astype(str).str.strip().str.replace(
22
23 # Visualização inicial melhorada do gráfico
24 plt.figure(figsize=(14, 7))
25 plt.plot(df['Data'], df['Preço do Petroleo'], label="Preço do Petróleo", color='navy')
26
27 # Formatação do eixo Y: Função para formatação com separador de milhar ou com poucas c
28 def y_formatter(x, pos):
29     return f'{x:,.2f}'
30
31 plt.gca().yaxis.set_major_formatter(FuncFormatter(y_formatter))
32 plt.xlabel("Data", fontsize=12)
33 plt.ylabel("Preço (US$)", fontsize=12)
34 plt.title("Variação Histórica do Preço do Petróleo", fontsize=14)
35 plt.legend(fontsize=12)
36 plt.tight_layout()
37 plt.show()
```



✓ 2. Seleção do Período e Normalização

- Para evitar problemas de overfitting e trabalhar com dados mais recentes, defini uma data de corte (por exemplo, a partir de 06/01/2020).
- Normalização dos preços utilizando o `MinMaxScaler`, transformando os valores para a escala $[0,1]$.

```
1 # 2. Seleção do Período e Normalização dos Dados
2 data_cut = pd.to_datetime("2020-06-01")
3 df_filtered = df[df['Data'] >= data_cut].copy()
4 df_filtered.reset_index(drop=True, inplace=True)
5
6 prices = df_filtered['Preço do Petróleo'].values.reshape(-1, 1)
```

```
7 scaler = MinMaxScaler()
8 prices_normalized = scaler.fit_transform(prices)
```

✓ 3. Criação das Sequências para o Modelo LSTM

- Definição de uma função `create_sequences` que cria sequências de entrada (X) e o próximo valor como saída (y) com base em um tamanho de sequência definido (ex.: 10).
- Separação dos dados normalizados em conjuntos de treinamento (80%) e teste (20%).

```
1 # Criando sequências
2 def create_sequences(data, sequence_length):
3     X = []
4     y = []
5     for i in range(len(data) - sequence_length):
6         X.append(data[i:i+sequence_length])
7         y.append(data[i+sequence_length])
8     return np.array(X), np.array(y)
9
10 # Definindo o comprimento da sequência
11 sequence_length = 10
12
13 # Criação das sequências
14 X, y = create_sequences(prices_normalized, sequence_length)
15
16 # Divisão em conjuntos de treino e teste (80% treino, 20% teste)
17 split = int(0.8 * len(X))
18 X_train, X_test = X[:split], X[split:]
19 y_train, y_test = y[:split], y[split:]
```

✓ 4. Construção e Treinamento do Modelo LSTM

- Criação de um modelo sequencial com uma camada LSTM contendo 350 unidades e uma camada densa final com 1 neurônio.
- Compilação do modelo com o otimizador Adam e a função de perda MSE. Treinamento do modelo por 100 épocas com tamanho de batch igual a 64.

```
1 from tensorflow.keras.models import Sequential, load_model
2 from tensorflow.keras.layers import LSTM, Dense
3 from tensorflow.keras.optimizers import Adam
4
5 # Construção do modelo LSTM
6 model = Sequential()
7 model.add(LSTM(350, input_shape=(sequence_length, 1)))
8 model.add(Dense(1))
9 model.compile(optimizer=Adam(), loss='mse')
10
11 # Treinamento do modelo
```

```
12 history = model.fit(X_train, y_train, epochs=100, batch_size=64, validation_data=(X_te
13
14 # Salvando o modelo treinado
15 model.save("meu_modelo.h5")
16
17 # Carregando o modelo
18 model = load_model("meu_modelo.h5", compile=False)
19
```



```

15/15 ----- 1s /4ms/step - loss: 5.7701e-04 - val_loss: 2.6464e-04
Epoch 98/100
15/15 ----- 1s 71ms/step - loss: 6.3692e-04 - val_loss: 4.0988e-04
Epoch 99/100
15/15 ----- 1s 71ms/step - loss: 6.7349e-04 - val_loss: 3.2134e-04
Epoch 100/100
15/15 ----- 1s 70ms/step - loss: 5.7099e-04 - val_loss: 4.2613e-04

```

✓ 5. Avaliação do Modelo

- Previsão dos valores de teste utilizando o modelo treinado.
- Cálculo das métricas de desempenho: R^2 , MSE, MAE, MAPE e RMSE.

```

1 # Previsões no conjunto de teste
2 y_pred_normalized = model.predict(X_test)
3 y_pred = scaler.inverse_transform(y_pred_normalized)
4 y_test_actual = scaler.inverse_transform(y_test)
5
6 # Cálculo das métricas
7 r2 = r2_score(y_test_actual, y_pred)
8 mse = mean_squared_error(y_test_actual, y_pred)
9 mae = mean_absolute_error(y_test_actual, y_pred)
10 mape = np.mean(np.abs((y_test_actual - y_pred) / y_test_actual)) * 100
11 rmse = math.sqrt(mse)
12
13 # Impressão das métricas
14 print(f"R² Score: {r2:.4f}")
15 print(f"MSE: {mse:.4f}")
16 print(f"MAE: {mae:.4f}")
17 print(f"MAPE: {mape:.4f}%")
18 print(f"RMSE: {rmse:.4f}")

```

```

↔ 8/8 ----- 0s 42ms/step
R² Score: 0.8693
MSE: 3.9970
MAE: 1.6485
MAPE: 2.0504%
RMSE: 1.9993

```

✓ 6. Previsão Futura e Visualização

- Definição de funções para prever os próximos pontos da série (função `predict_future`) e gerar as datas correspondentes.
- Visualização do histórico filtrado juntamente com as previsões para os próximos 15 dias.

```

1 # Função para realizar previsões futuras a partir dos últimos dados conhecidos
2 def predict_future(model, data, num_prediction, sequence_length, scaler):
3     # Converte os últimos sequence_length pontos para float para garantir homogeneidad
4     prediction_list = [float(item) for item in data[-sequence_length:]]

```

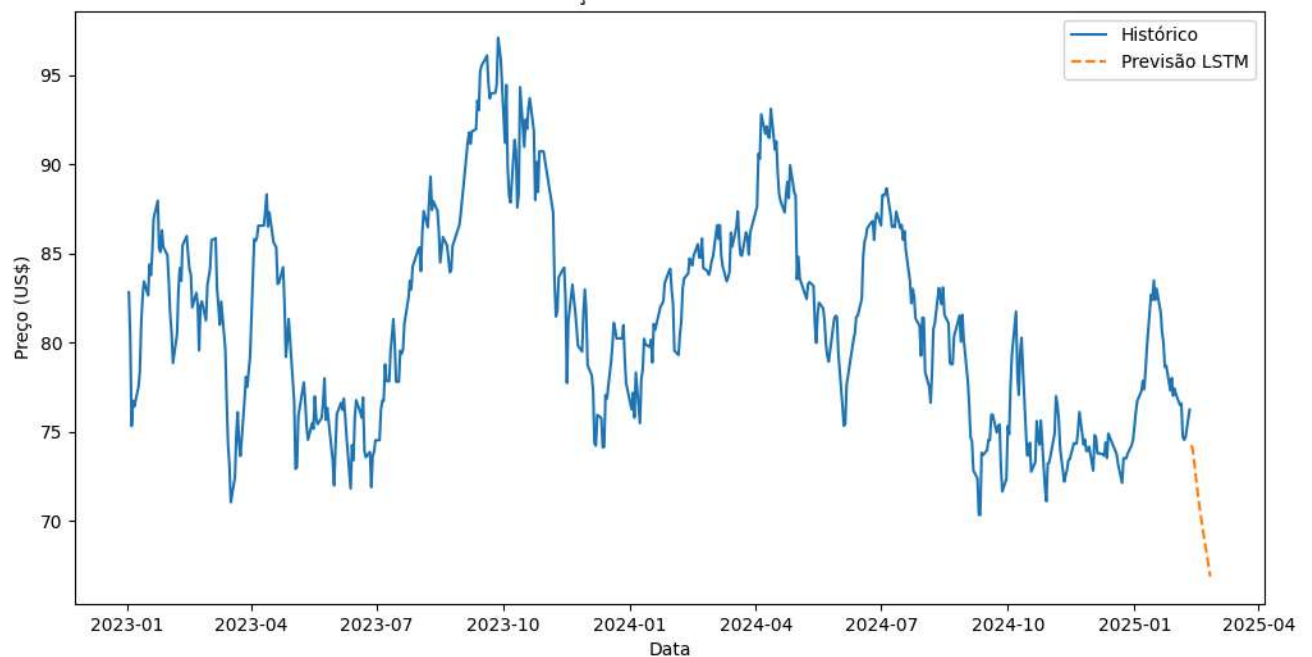
```
5
6     for _ in range(num_prediction):
7         # Converte a lista para array NumPy e reshape para (1, sequence_length, 1)
8         x = np.array(prediction_list[-sequence_length:], dtype=float).reshape((1, sequ
9         out = model.predict(x)[0][0]
10        prediction_list.append(out)
11    # Seleciona apenas os valores previstos (excluindo os dados de entrada)
12    prediction_list = prediction_list[sequence_length:]
13    # Desnormaliza as previsões
14    prediction_list = scaler.inverse_transform(np.array(prediction_list).reshape(-1, 1
15    return prediction_list
16
17 # Função para gerar datas futuras com base na última data do DataFrame
18 def predict_dates(last_date, num_prediction):
19     return pd.date_range(start=last_date + datetime.timedelta(days=1), periods=num_pre
20
21 # Definindo o número de dias a serem previstos
22 num_prediction = 15
23 forecast = predict_future(model, prices_normalized, num_prediction, sequence_length, s
24 forecast_dates = predict_dates(df_filtered['Data'].iloc[-1], num_prediction)
25
26 # Preparação para plotagem: dados históricos a partir de uma data de corte para melhor
27 data_cut_plot = pd.to_datetime("2023-01-01")
28 df_plot = df_filtered[df_filtered['Data'] >= data_cut_plot].copy()
29
30 # Plotando os dados históricos e a previsão futura
31 plt.figure(figsize=(12,6))
32 plt.plot(df_plot['Data'], df_plot['Preço do Petroleo'], label="Histórico")
33 plt.plot(forecast_dates, forecast, label="Previsão LSTM", linestyle="--")
34 plt.xlabel("Data")
35 plt.ylabel("Preço (US$)")
36 plt.title("Previsão do Preço do Petróleo com Modelo LSTM")
37 plt.legend()
38 plt.show()
```

```

1/1 ██████████ 0s 43ms/step
1/1 ██████████ 0s 20ms/step<ipython-input-45-025e085bc3ec>:4: DeprecationWarning
    prediction_list = [float(item) for item in data[-sequence_length:]]
1/1 ██████████ 0s 43ms/step
1/1 ██████████ 0s 39ms/step
1/1 ██████████ 0s 39ms/step
1/1 ██████████ 0s 37ms/step
1/1 ██████████ 0s 47ms/step
1/1 ██████████ 0s 37ms/step
1/1 ██████████ 0s 40ms/step
1/1 ██████████ 0s 53ms/step
1/1 ██████████ 0s 37ms/step
1/1 ██████████ 0s 42ms/step
1/1 ██████████ 0s 49ms/step
1/1 ██████████ 0s 45ms/step
1/1 ██████████ 0s 54ms/step
1/1 ██████████ 0s 45ms/step

```

Previsão do Preço do Petróleo com Modelo LSTM



```

1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from matplotlib.ticker import FuncFormatter
5 import datetime
6
7 # Definindo o número de dias a serem previstos (ex.: 15 dias)
8 num_prediction = 15

```

```
9 forecast = predict_future(model, prices_normalized, num_prediction, sequence_length, s
10 forecast_dates = predict_dates(df_filtered['Data'].iloc[-1], num_prediction)
11
12 # Converter as previsões para um DataFrame
13 df_forecast = pd.DataFrame({
14     'Data': forecast_dates,
15     'Preço do Petroleo': forecast.flatten(),
16     'Tipo': 'Previsão'
17 })
18
19 # Selecionar o período de interesse: de 01/01/2025 até 25/02/2025 (por exemplo)
20 start_zoom = pd.to_datetime("2025-01-01")
21 end_zoom = pd.to_datetime("2025-02-25")
22
23 # Dados históricos (já existentes) dentro do período de zoom
24 df_historico_zoom = df_filtered[(df_filtered['Data'] >= start_zoom) & (df_filtered['Da
25 df_historico_zoom['Tipo'] = 'Histórico'
26
27 # Combinar os dados históricos e as previsões que caem no período de zoom
28 # Algumas previsões podem estar fora do período; vamos filtrar as previsões também.
29 df_forecast_zoom = df_forecast[(df_forecast['Data'] >= start_zoom) & (df_forecast['Dat
30
31 # Concatenar os DataFrames para criar uma tabela completa
32 df_zoom = pd.concat([df_historico_zoom, df_forecast_zoom]).sort_values('Data').reset_i
33
34 # Gráfico: zoom no período de janeiro e fevereiro de 2025
35 plt.figure(figsize=(14, 7))
36
37 # Plotar os dados históricos
38 plt.plot(df_historico_zoom['Data'], df_historico_zoom['Preço do Petroleo'], label="His
39
40 # Plotar as previsões
41 plt.plot(df_forecast_zoom['Data'], df_forecast_zoom['Preço do Petroleo'], label="Previ
42
43 # Formatação do eixo Y para facilitar a leitura
44 def y_formatter(x, pos):
45     return f'{x:,.2f}'
46 plt.gca().yaxis.set_major_formatter(FuncFormatter(y_formatter))
47
48 plt.xlabel("Data", fontsize=12)
49 plt.ylabel("Preço (US$)", fontsize=12)
50 plt.title("Zoom: Preço do Petróleo (Jan - Fev/2025)", fontsize=14)
51 plt.legend(fontsize=12)
52 plt.tight_layout()
53 plt.show()
54
55 # Exibindo a tabela dos dados (históricos + previsão) no período selecionado
56 print("Tabela de Dados (Histórico + Previsão) para o Período de 01/01/2025 a 25/02/202
57 print(df_zoom[['Data', 'Preço do Petroleo', 'Tipo']].to_string(index=False))
58
```



```
1/1 ██████████ 0s 50ms/step
1/1 ██████████ 0s 24ms/step<ipython-input-45-025e085bc3ec>:4: DeprecationWa
    prediction_list = [float(item) for item in data[-sequence_length:]]
1/1 ██████████ 0s 47ms/step
1/1 ██████████ 0s 47ms/step
1/1 ██████████ 0s 45ms/step
1/1 ██████████ 0s 74ms/step
1/1 ██████████ 0s 72ms/step
1/1 ██████████ 0s 76ms/step
1/1 ██████████ 0s 64ms/step
1/1 ██████████ 0s 68ms/step
1/1 ██████████ 0s 61ms/step
1/1 ██████████ 0s 64ms/step
1/1 ██████████ 0s 62ms/step
1/1 ██████████ 0s 84ms/step
1/1 ██████████ 0s 63ms/step
1/1 ██████████ 0s 75ms/step
```

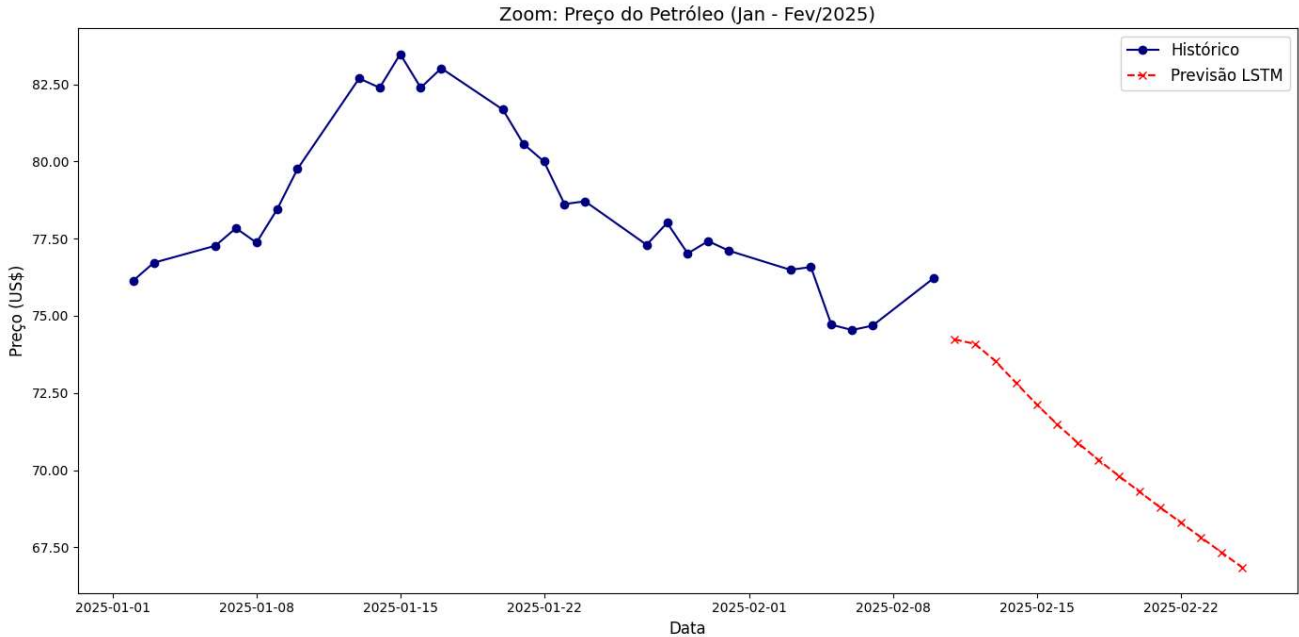


Tabela de Dados (Histórico + Previsão) para o Período de 01/01/2025 a 25/02/2025:

Data	Preco do Petroleo	Tipo
2025-01-02	76.140000	Histórico
2025-01-03	76.720000	Histórico
2025-01-06	77.270000	Histórico
2025-01-07	77.840000	Histórico
2025-01-08	77.370000	Histórico
2025-01-09	78.440000	Histórico
2025-01-10	79.760000	Histórico
2025-01-13	82.690000	Histórico
2025-01-14	82.390000	Histórico
2025-01-15	83.480000	Histórico
2025-01-16	82.390000	Histórico
2025-01-17	83.020000	Histórico
2025-01-20	81.680000	Histórico
2025-01-21	80.570000	Histórico
2025-01-22	80.000000	Histórico
2025-01-23	78.620000	Histórico
2025-01-24	78.710000	Histórico
2025-01-27	77.300000	Histórico
2025-01-28	78.010000	Histórico
2025-01-29	77.020000	Histórico
2025-01-30	77.420000	Histórico
2025-01-31	77.110000	Histórico
2025-02-01	74.200000	Previsão LSTM
2025-02-02	73.800000	Previsão LSTM
2025-02-03	73.400000	Previsão LSTM
2025-02-04	73.000000	Previsão LSTM
2025-02-05	72.600000	Previsão LSTM
2025-02-06	72.200000	Previsão LSTM
2025-02-07	71.800000	Previsão LSTM
2025-02-08	71.400000	Previsão LSTM
2025-02-09	71.000000	Previsão LSTM
2025-02-10	70.600000	Previsão LSTM
2025-02-11	70.200000	Previsão LSTM
2025-02-12	69.800000	Previsão LSTM
2025-02-13	69.400000	Previsão LSTM
2025-02-14	69.000000	Previsão LSTM
2025-02-15	68.600000	Previsão LSTM
2025-02-16	68.200000	Previsão LSTM
2025-02-17	67.800000	Previsão LSTM
2025-02-18	67.400000	Previsão LSTM
2025-02-19	67.000000	Previsão LSTM
2025-02-20	66.600000	Previsão LSTM
2025-02-21	66.200000	Previsão LSTM
2025-02-22	65.800000	Previsão LSTM
2025-02-23	65.400000	Previsão LSTM
2025-02-24	65.000000	Previsão LSTM
2025-02-25	64.600000	Previsão LSTM

2025-02-03	76.490000	Histórico
2025-02-04	76.580000	Histórico
2025-02-05	74.710000	Histórico
2025-02-06	74.540000	Histórico
2025-02-07	74.680000	Histórico
2025-02-10	76.230000	Histórico
2025-02-11	74.233444	Previsão
2025-02-12	74.088318	Previsão
2025-02-13	73.522041	Previsão
2025-02-14	72.820374	Previsão
2025-02-15	72.125267	Previsão
2025-02-16	71.481239	Previsão
2025-02-17	70.883293	Previsão
2025-02-18	70.334122	Previsão
2025-02-19	69.809425	Previsão
2025-02-20	69.302811	Previsão
2025-02-21	68.795914	Previsão
2025-02-22	68.309334	Previsão
2025-02-23	67.819733	Previsão
2025-02-24	67.333382	Previsão
2025-02-25	66.851944	Previsão