- Tech Challenge 2

Introdução

A previsão de séries temporais desempenha um papel crucial em diversos setores da economia, com destaque para o mercado financeiro. A capacidade de prever o fechamento da bolsa de valores, em particular, é de grande importância para investidores, analistas e gestores, pois oferece suporte para tomadas de decisão mais embasadas e estratégicas.

O fechamento da bolsa de valores reflete um conjunto de fatores econômicos, políticos e sociais que influenciam o comportamento dos preços de ativos ao longo do tempo. Nesse cenário dinâmico e volátil, métodos tradicionais de análise, como os fundamentados exclusivamente em histórico de preços, podem não capturar a complexidade inerente das interações de mercado.

Neste contexto, o uso de modelos de previsão de séries temporais surge como uma solução poderosa. Modelos como ARIMA (Autoregressive Integrated Moving Average), SARIMA (Seasonal ARIMA) e PROPHET, têm demonstrado elevado potencial em capturar padrões temporais complexos e prever valores futuros com precisão.

O analista de dados desempenha um papel fundamental nesse processo, sendo responsável por coletar, organizar e explorar os dados relevantes, garantindo que estejam adequados para a modelagem. Além disso, é o analista que avalia diferentes modelos preditivos, seleciona as abordagens mais adequadas e interpreta os resultados, traduzindo-os em insights valiosos para os tomadores de decisão. Sua expertise é essencial para assegurar que as previsões sejam robustas, confiáveis e alinhadas com os objetivos do negócio.

O objetivo deste estudo é desenvolver e avaliar um modelo de previsão capaz de estimar o fechamento da bolsa de valores a partir de dados históricos com uma assertividade de 70%.

Sobre os dados

Na construção do modelo preditivo, utilizaremos os dados disponíveis no site Investing como fonte principal. A tabela obtida apresenta as seguintes colunas, que serão analisadas para medir o desempenho diário do Ibovespa:

- Total de Registros: 2727 entradas, cobrindo o período de janeiro de 2014 a dezembro de 2024

- Data: Refere-se ao dia específico da cotação, permitindo identificar o período exato da análise.

- Último: Apresenta o valor de fechamento do índice no final do pregão, ou seja, o último preço registrado quando o mercado encerra suas atividades naquele dia.

- Abertura: Indica o valor inicial do índice no início do pregão daquele dia.

- Máxima: Mostra o valor mais alto atingido pelo índice durante o pregão, evidenciando o pico de valorização ao longo do dia.

- Mínima: Aponta o valor mais baixo alcançado pelo índice durante o período de negociação.

- Volume: Refere-se ao volume financeiro total negociado no mercado durante o dia, indicando a intensidade da negociação.

- Variação (%): Exibe a variação percentual do índice em relação ao dia anterior, permitindo avaliar se houve valorização ou desvalorização no período.

Processamento e tratamento dos dados

Para a construção de um modelo o primeiro passo é o tratamento da base e para este projeto os passos foram os seguintes:

- Data foi convertida para o tipo datetime

```
df_ibovespa['Data'] = pd.to_datetime(df_ibovespa['Data'], format='%d.%m.%Y')
```

- Vol. agora é um número em notação decimal, lidando adequadamente com valores em milhões e milhares.

```
def convert_volume(volume_str):
    if isinstance(volume_str, str):
        if 'M' in volume_str:
            return float(volume_str.replace('M', '').replace(',', '.')) * 1_000_000
        elif 'K' in volume_str:
            return float(volume_str.replace('K', '').replace(',', '.')) * 1_000
        return float(volume_str.replace(',', '.'))
    return volume_str

df_ibovespa['Vol.'] = df_ibovespa['Vol.'].apply(convert_volume)
```

- Var% também foi convertida para número decimal

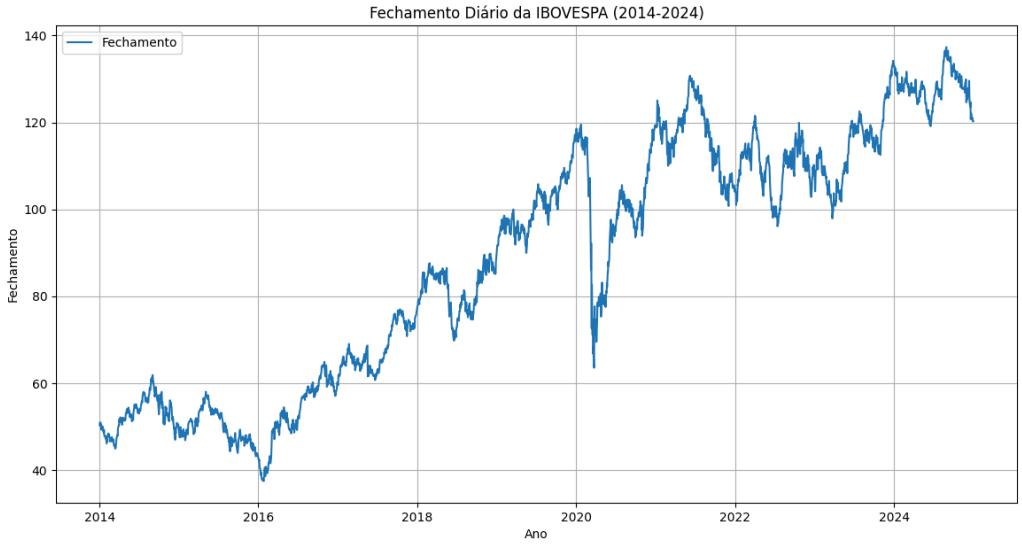
  df_ibovespa['Var%'] = df_ibovespa['Var%'].str.replace('%', '').str.replace(',', '.').astype(float)

  .
- Atribuindo o campo de data como index.

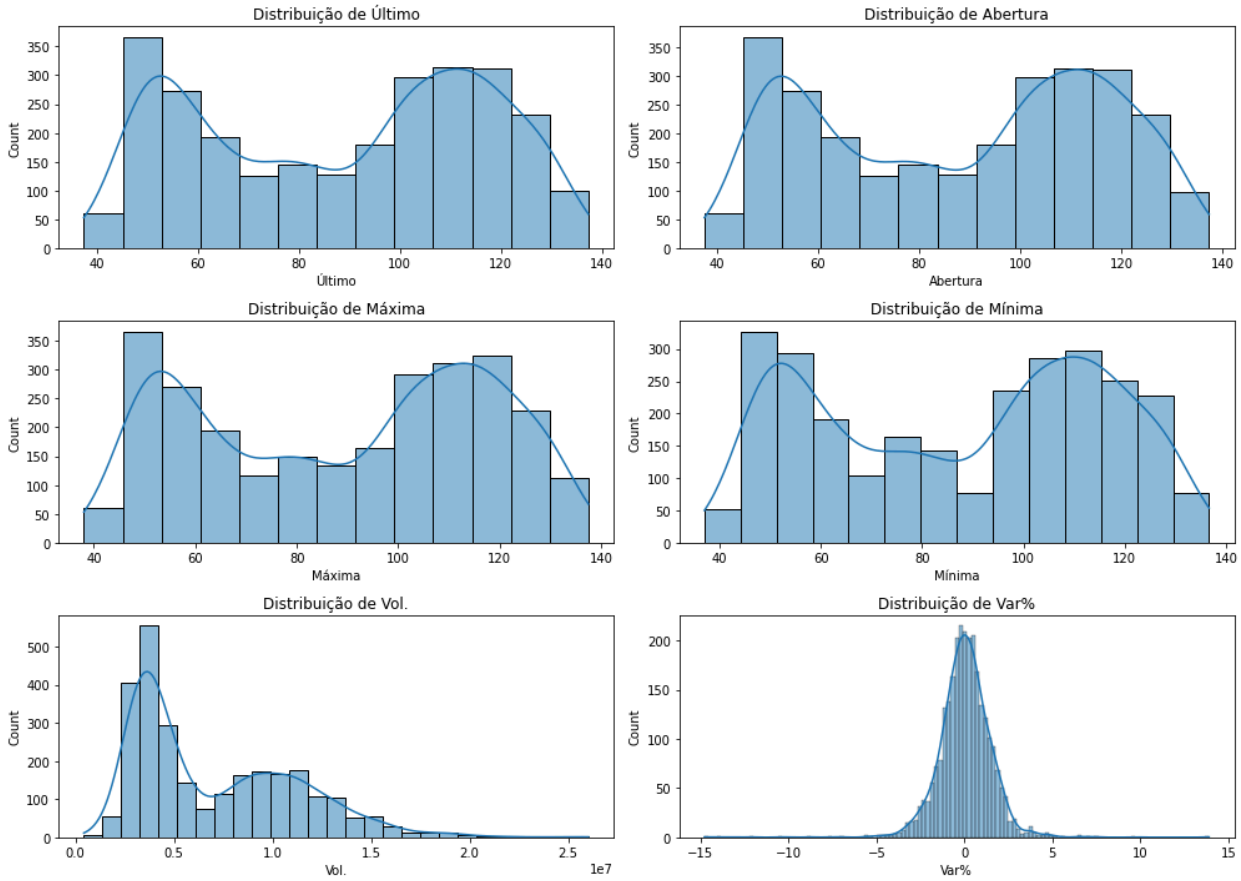
  df_ibovespa.set_index('Data')['Último']

- Informação ausente: Há um único valor ausente na coluna Vol., que poderemos tratar nas etapas posteriores, dependendo de como ele afetará a modelagem

Estatísticas de introdução e histogramas

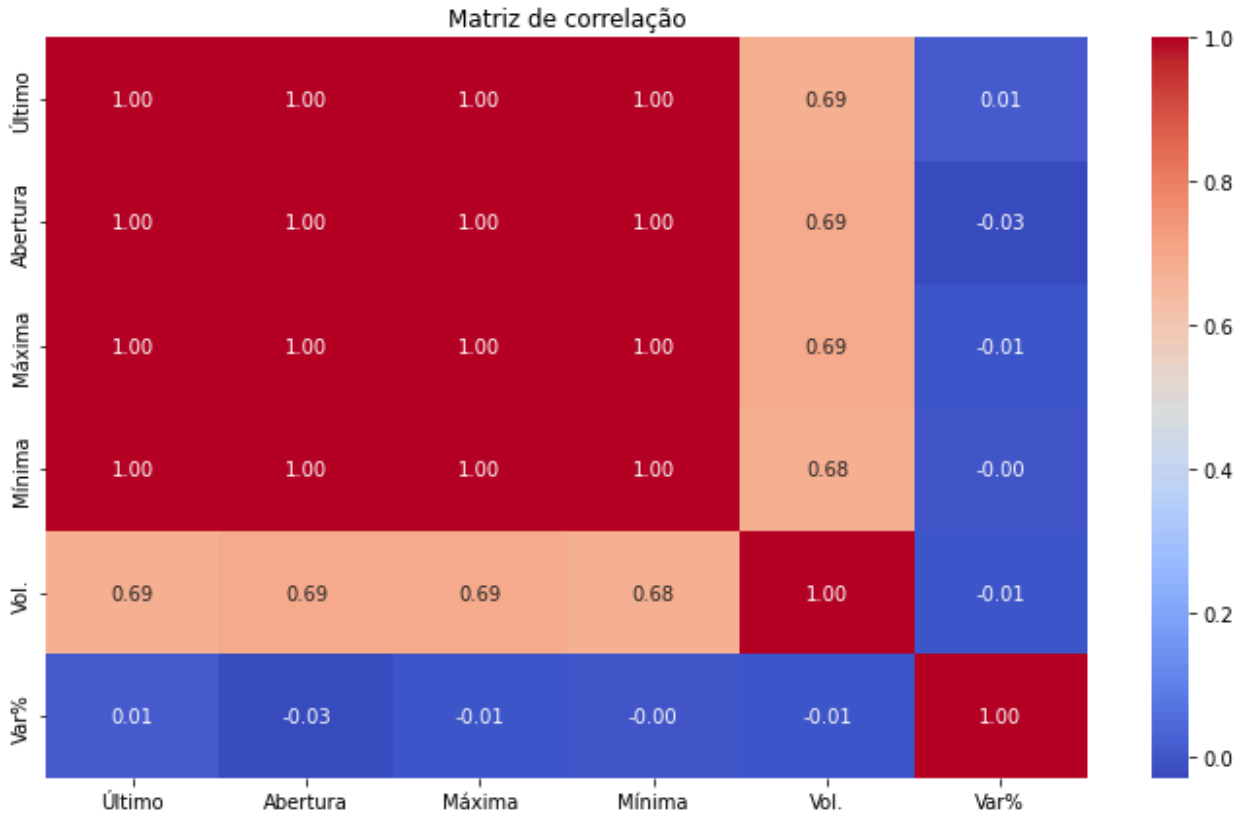


Há uma tendência geral de alta no índice IBOVESPA na última década, indicando crescimento no mercado. No entanto, essa tendência é acompanhada por flutuações significativas e períodos de aumento e declínio. O mercado exibe considerável volatilidade, o que é típico para índices de ações. Há picos e quedas visíveis, refletindo reações a eventos econômicos, mudanças de política ou influências do mercado global.



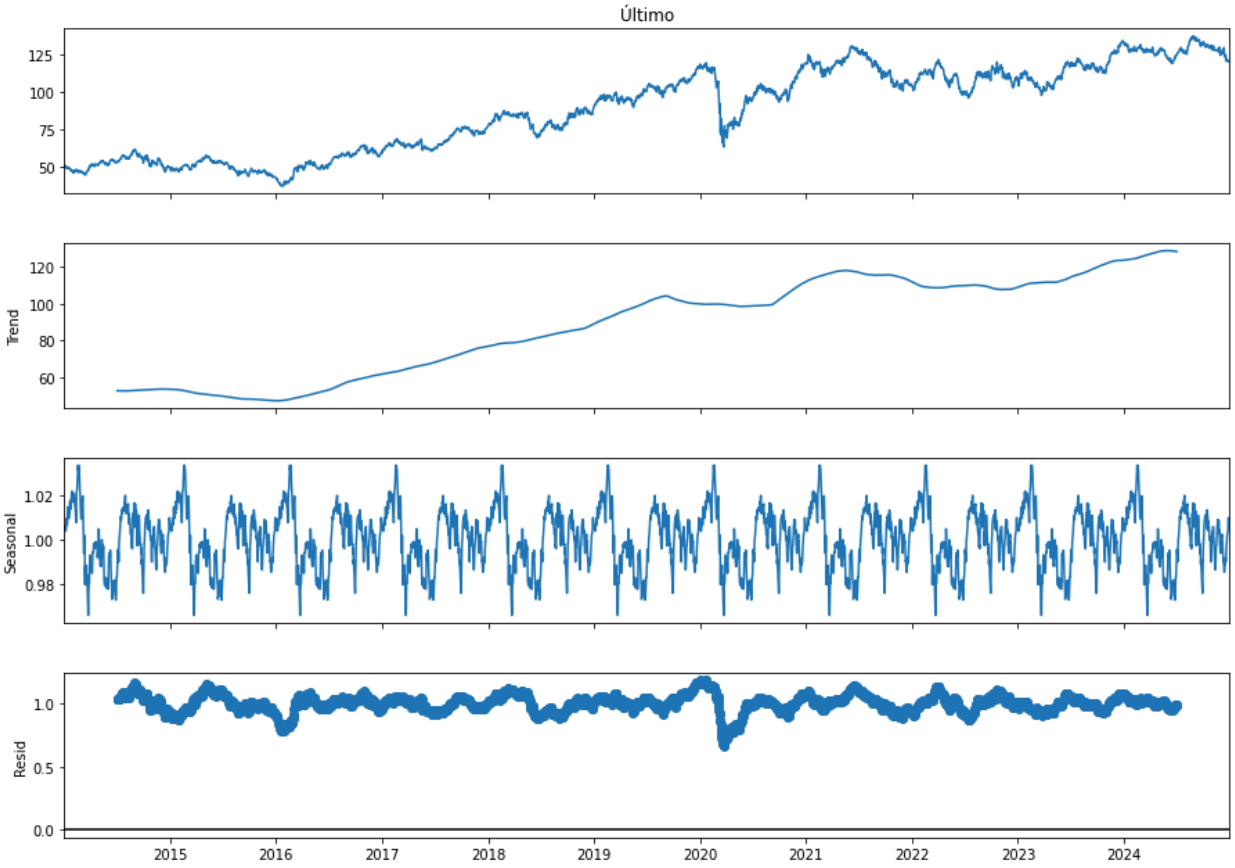
As distribuições exibem padrões interessantes: os preços de fechamento, abertura, máximos e mínimos apresentam distribuições bimodais, sugerindo concentração em duas faixas de valores. O volume de negociação mostra assimetria, com maior frequência em volumes baixos e uma cauda longa à direita. Já a variação percentual tem uma distribuição próxima do normal, concentrada em torno de zero, indicando estabilidade na maioria dos casos.

Regressão (Correlação das métricas)



A matriz de correlação revela forte correlação positiva (valor 1.00) entre os preços de Último, Abertura, Máxima e Mínima, indicando que eles variam de forma muito semelhante. O Volume (Vol.) tem correlação moderada (~0.69) com esses preços, sugerindo relação menos direta. Já a variação percentual (Var%) tem correlação praticamente nula com os demais fatores, indicando independência em relação a preços e volume.

Análise temporal



A decomposição da série mostra:

1. **Série Original**: Tendência geral de alta ao longo do tempo.
2. **Tendência**: Movimento ascendente contínuo com algumas oscilações leves.
3. **Sazonalidade**: Padrões regulares recorrentes em ciclos curtos.
4. **Resíduo**: Ruído aparentemente aleatório, sem padrão definido.

Isso sugere que a série é influenciada por tendência e sazonalidade, com flutuações residuais não explicadas.

A presença de uma sazonalidade por si só não implica que a série seja estacionária. Para uma série ser estacionária, sua média, variância e autocorrelação devem ser constantes ao longo do tempo.

No gráfico, há uma **tendência crescente** evidente, o que viola a condição de estacionariedade. Além disso, a sazonalidade indica padrões cíclicos, mas a presença de uma tendência sugere que a série não é estacionária sem um ajuste prévio, como a remoção da tendência (diferenciação, por exemplo).

Testes de estacionaridade

Usaremos o teste **Augmented Dickey-Fuller (ADF)**, para verificar se os dados são estacionários.

```
# ADF test function

def adf_test(series):

    result = adfuller(series, autolag='AIC')

    return {'ADF_Statistic': result[0], 'p-value': result[1], 'Critical_Values': result[4]}


daily_adf_result = adf_test(daily_data)

print(f"ADF_Statistic: {daily_adf_result['ADF_Statistic']}")

print(f"p-value: {daily_adf_result['p-value']}")


print('Critical Values:')

for key, value in daily_adf_result['Critical_Values'].items():

    print('\t%s: %.3f' % (key, value))
```

H0: Hipótese nula - A série é não estacionária

H1: Rejeição de H0 - A série é estacionária

O teste ADF foi realizado nos dados diários e chegamos nesses resultados:

```
ADF_Statistic: -1.4447455372713434

p-value: 0.5606466715296446
```

Com um p-valor de 0,561, não rejeitamos a hipótese nula (H0), ou seja os dados não são estacionários.

Para tornar os dados estacionários e poder usar o ARIMA para predição de valores futuros, aplicaremos a diferenciação e verificaremos novamente.

Após a diferenciação, tivemos os seguintes resultados:

ADF_Statistic: -13.831187405458527
p-value: 7.587161234013877e-26
Critical Values:
1%: -3.432
5%: -2.862
10%: -2.567

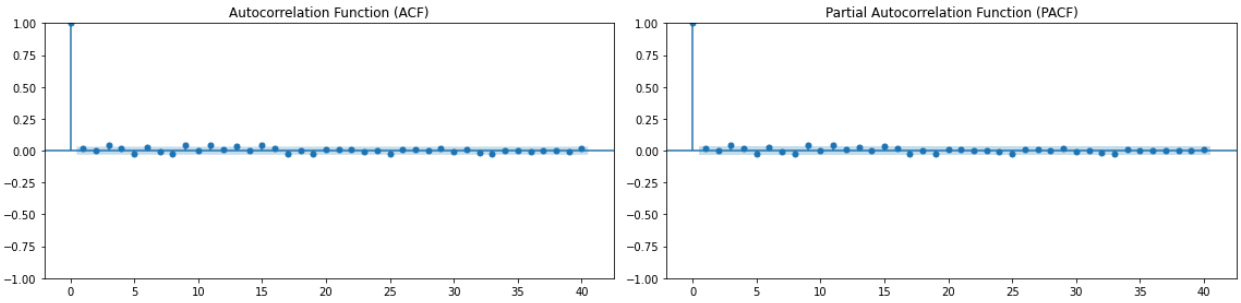
Com um p-valor extremamente baixo, podemos rejeitar H0. Portanto, os dados diferidos são estacionários. Deste modo, podemos prosseguir com a modelagem com ARIMA.

Quando construímos um modelo ARIMA, temos que considerar os termos p , d e q que entram em nosso modelo ARIMA.

O primeiro parâmetro, p , é o número de observações defasadas. Ao considerar p , determinamos efetivamente o quão longe no tempo voltamos ao tentar prever a observação atual. Fazemos isso observando as auto correlações de nossa série temporal, que são as correlações em nossa série em defasagens de tempo anteriores.

O segundo parâmetro, d, refere-se à ordem de diferenciação, sobre a qual falamos. Novamente, a diferenciação significa simplesmente encontrar as diferenças entre intervalos de tempo consecutivos. É uma maneira de tornar nossos dados estacionários, o que significa remover as tendências e quaisquer mudanças na variância ao longo do tempo indica a diferenciação em qual ordem você obtém um processo estacionário.

O terceiro parâmetro se refere à ordem da parte da média móvel (MA) do modelo. Ele representa o número de erros de previsão defasados incluídos no modelo. Diferentemente de uma média móvel simples, que suaviza os dados, a média móvel no ARIMA captura a relação entre uma observação e os erros residuais de um modelo de média móvel aplicado a observações defasadas.



Com base nestes gráficos, podemos ver que o ACF corta a linha pontilhada azul na lag 1, enquanto o PACF corta a linha pontilhada azul na lag 1. Isso sugere que um modelo ARIMA(1,1,1) pode ser adequado para a série temporal já que fizemos 1 diferenciação.

Projeção 1 - ARIMA

Usamos 9 dos 10 anos para treinamento e deixamos 1 ano de dados para validação. Segue as previsões para o ano de 2024. Podemos comparar com os dados reais e ver se a estimativa está boa.

```
# Modelagem com ARIMA
closing_prices = df_ibovespa.set_index('Data')['Último'].sort_index()
train_data = closing_prices[:'2023-12-31']
test_periods = pd.date_range('2024-01-01', periods=365, freq='B')

model_arima = ARIMA(train_data, order=(1,1,1))
model_fit = model_arima.fit()

forecast_arima = model_fit.get_forecast(steps=len(test_periods))
forecast_values = forecast_arima.predicted_mean
forecast_values_lower = forecast_arima.conf_int(alpha=0.05).iloc[:, 0]
forecast_values_upper = forecast_arima.conf_int(alpha=0.05).iloc[:, 1]

df_forecast = pd.DataFrame({
    'Data': test_periods,
    'Previsão': forecast_values,
    'Intervalo Inferior': forecast_values_lower,
    'Intervalo Superior': forecast_values_upper
})

—----------------------------------------------------------------------------------------------
# Plotando os resultados
plt.figure(figsize=(12, 6))

# Dados históricos
plt.plot(train_data.index, train_data, label='Dados Treinamento', color='blue')
plt.plot(closing_prices['2023-12-31':].index, closing_prices['2023-12-31':], label='Dados de teste',
color='green')
# Previsões com intervalos de confiança
plt.plot(df_forecast['Data'], df_forecast['Previsão'], label='Previsão', color='orange')
plt.fill_between(
    df_forecast['Data'],
    df_forecast['Intervalo Inferior'],
    df_forecast['Intervalo Superior'],
    color='orange', alpha=0.2, label='Intervalo de Confiança'
)

# Adicionando rótulos e título
plt.title('Previsão de Fechamento com ARIMA', fontsize=14)
plt.xlabel('Data', fontsize=12)
plt.ylabel('Preço de Fechamento', fontsize=12)
plt.legend()
```

```
plt.grid()
```

Calculamos o RMSE do modelo e chegamos em um resultado de 7,06. Ou seja, o RMSE de 7,06 significa que, em média, as previsões do nosso modelo desviam dos preços reais em cerca de R$7,06.

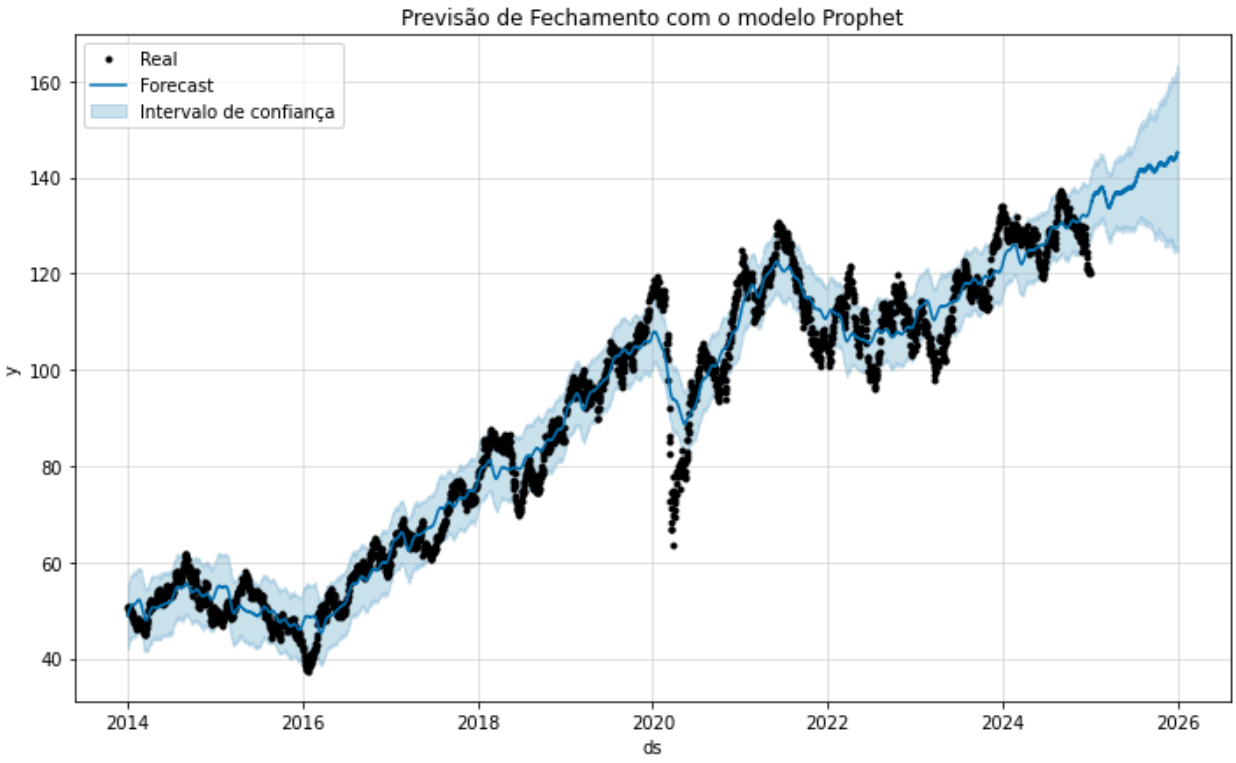
```
forecast_ar = forecast_values[:len(closing_prices['2023-12-31':])]
test_close = closing_prices['2023-12-31':]

# Calculate RMSE
rmse = np.sqrt(mean_squared_error(test_close, forecast_ar))
```

Projeção 2 - Prophet

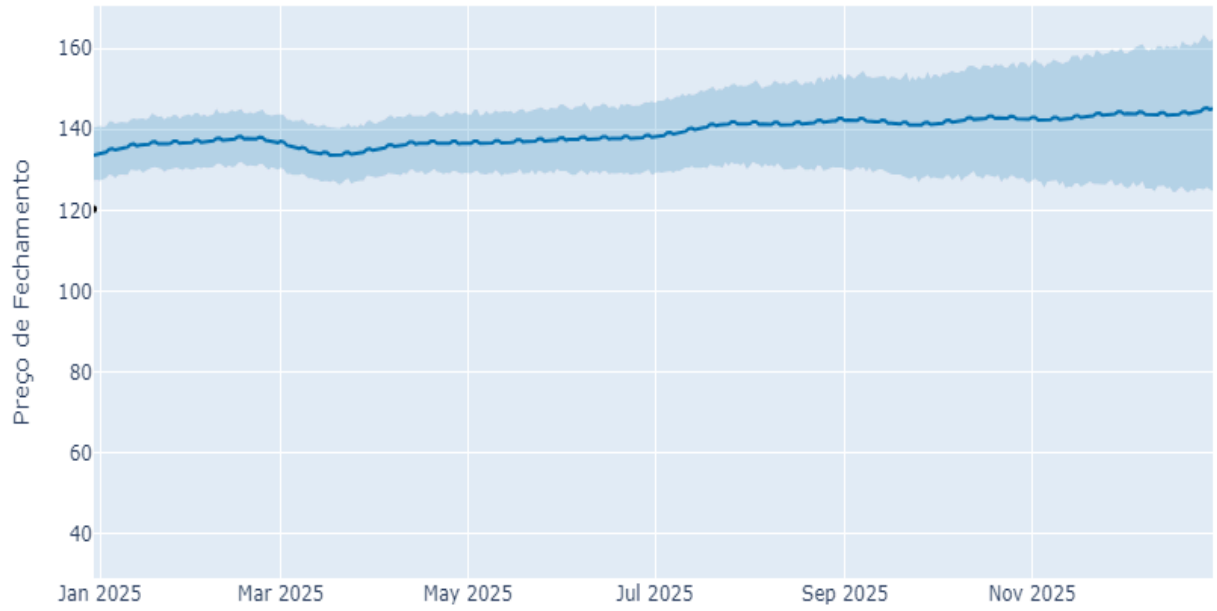
Previsões para o ano de 2025. O modelo parece ter capturado melhor a complexidade da série temporal. Podemos observar o ajuste no gráfico abaixo. De acordo com o modelo, fecharemos o ano de 2025 com um valor de ~145.

```
m = Prophet()
m.fit(model_df)
future = m.make_future_dataframe(periods=365)
forecast = m.predict(future)
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
```

```
fig = m.plot(forecast)
ax = fig.gca()
ax.set_title('Previsão de Fechamento com o modelo Prophet')
ax.legend(['Real', 'Forecast', 'Intervalo de confiança'], loc='upper left')
plt.show()
```



Zoom na curva do forecast



Seguem as métricas de performance do modelo para diferentes horizontes de tempo (forecast)

horizon	mse	rmse	mae	mape	mdape	smape	coverage
36 days	67.196240	8.197331	6.437001	0.066832	0.054870	0.068016	0.384996
37 days	67.274062	8.202077	6.471858	0.067280	0.055044	0.068436	0.379256
38 days	67.050895	8.188461	6.483300	0.067543	0.055385	0.068702	0.378266
39 days	65.895052	8.117577	6.443388	0.067534	0.055385	0.068760	0.380048
40 days	64.966843	8.060201	6.428958	0.067826	0.055468	0.069145	0.377276

Conclusão