PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA
Pós-graduação *Lato Sensu* em Ciência de Dados e Big Data

Sandy Yumi Delvalle

AVALIAÇÃO DO IMPACTO DE VARIÁVEIS MACROECONÔMICAS NO ÍNDICE BOVESPA

Belo Horizonte
2023

Sandy Yumi Delvalle

AVALIAÇÃO DO IMPACTO DE VARIÁVEIS MACROECONÔMICAS NO ÍNDICE BOVESPA

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Ciência de Dados e Big Data como requisito parcial à obtenção do título de especialista.

Belo Horizonte
2023

SUMÁRIO

Agradecimentos

Agradecimento a Deus pelo auxílio e proteção.

Agradecimentos especiais ao meu esposo Edson Carlos e a minha filha Melissa Mayumi, no qual apoiaram incondicionalmente a execução do meu trabalho, permitindo que enquanto eu estude, realizar as demais atividades, além das trocas diárias, convívio e discussões.

Agradecimento aos meus pais, irmãos, sogra, cunhados e familiares que criam o ambiente de segurança e proteção.

Agradecimento a rede de apoio (colegas de estudo e de trabalho).

Resumo

Este estudo tem como principal objetivo analisar a relação causal entre o conjunto de variáveis econômicas amplamente conhecidos, o IPCA, IGPM e a Selic e o mercado acionário brasileiro, representado exclusivamente pelo Índice Bovespa (ibov). Para tal, foi identificado a correlação das variáveis independentes (IPCA, IPGM e Selic) com a varável dependente (Ibov). O período analisado de todos os indicadores foi de Janeiro de 2010 a Abril de 2023 com frequência mensal. O modelo proposto foi identificar o quanto esses indicadores macroeconômicos podem explicar a variável Ibov e para tal, foram testados os métodos de Regressão Linear Múltipla, Análise de Série Temporal, Regressão Ridge, LSTM e Lasso. O resultado da regressão linear múltipla, Ridge e Lasso foram semelhantes. A análise de série temporal e o LSTM exibiram resultados abaixo dos demais, impossibilitando o uso desses métodos em previsão do Ibov. A inutilização da análise de série temporal é provavelmente explicada pelo fato da série ter se mostrado como não estacionária e possuir sazonalidades e outros fatores influenciando-a, como fatores políticos e condições econômicas do mundo, não apenas do próprio Ibov. Para as demais, chegou-se a explicar o IBOV pelos índices macroeconômicos em até 58%, no modelo de Regressão Ridge. Os resultados deixam claro que esses fatores podem influenciar o Ibov, porém existem muitos outros fatores que precisam ser adicionados para que tenha um bom modelo.

1. Introdução

1.1. Contextualização

O índice Bovespa (BOV: ^BVSP) é o principal índice da bolsa de valores brasileira, a B3 (Brasil, Bolsa, Balcão), que foi criado em 1968. Ele é composto pelas ações das empresas mais negociadas na B3 e é considerado um dos principais indicadores do mercado financeiro brasileiro. O índice Bovespa é calculado a partir da variação dos preços das ações das empresas componentes da sua carteira teórica, tendo como principais hoje, ações como Petrobras (PETR4 e PETR3), Vale (VALE3), Itaú Unibanco (ITUB4), Bradesco (BBDC4), B3 (B3SA3), Banco do Brasil (BBAS3), Ambev (ABEV3), Banco Bradesco (BBDC4), Santander (SANB11), porém, é importante ressaltar que o índice atualmente conta com mais de 70 empresas.

Quanto às principais variáveis que podem alterar o preço da ação das empresas que compõem o índice Bovespa, é importante destacar que diversos fatores podem influenciar o comportamento das ações na bolsa. Alguns dos principais fatores que podem afetar o preço das ações incluem:

a. Performance financeira da empresa;
b. Perspectivas de crescimento do setor em que a empresa atua;
c. Mudanças na política econômica e fiscal do governo;
d. Variações na taxa de juros;
e. Condições macroeconômicas do país e do mundo;
f. Instabilidade política;
g. Comportamento do mercado financeiro e dos investidores.

Mukherjee & Naka (1995) investigaram a relação entre o índice Tokyo Stoke Exchange (TSE) e outras seis variáveis econômicas: taxa de cambio, oferta de moeda, inflação, produção industrial, taxa de longo prazo de títulos e call Money rate com a metodologia VECM desenvolvida por Johansen (1991) e o resultados mostram uma relação negativa entre o índice TSE e a inflação e taxa de longo prazo dos títulos do governo; as demais apresentam uma relação positiva. Os resultados indicam que a inflação causa variações negativas no retorno acionário.

Além de Mukherjee & Naka, diversos autores explicam a variação dos indicadores macroeconômicos frente aos índices locais. O Quadro 1 apresenta um resumo não exaustivo quanto a literatura empírica:

Autor, Ano, País, Período, Método	**Variáveis**	**Resultado**
Geske e Roll, 1983, EUA, 1947-1980, Arima e MQO	1. Ganhos corporativos; 2. Taxa de desemprego; 3. Taxa de juros; 4. Inflação.	Um choque real (descontada a taxa de inflação do período) negativo aleatório afeta os retornos das ações, resultando em maior desemprego e menor ganhos coorporativos
Mukherjee & Darrat, 1986, Índia, 1948-1984, VAR	1. Oferta monetária; 2. Taxa de juros de curto prazo e longo prazo; 3. PNB per capita; 4. Inflação.	Há defasagem significativa entre os retornos dos ativos e o crescimento da oferta monetária; Há relação negativa entre retorno dos ativos e taxa de juros de longo prazo e inflação
Mukherjee & Naka, 1995, Japão, 1971-1990, VEC	1. Taxa de câmbio; 2. Oferta de moeda; 3. Inflação; 4. Produção industrial; Taxa de longo prazo dos títulos do governo.	Há relação negativa entre o mercado acionário, a taxa de inflação e a taxa de longo prazo dos títulos do governo; Há relação positiva entre o mercado acionário e a taxa de câmbio, oferta de moeda, produção industrial e call money rate
Gjerde e Saettem, 1999, Noruega, 1974-1994, VAR e Causalidade de Granger	1. Produção industrial; 2. Consumo; 3. Inflação; 4. Taxa de câmbio; 5. Preço do barril do petróleo; 6. Taxa de juros; 7. Índice de produção industrial OCDE.	Os retornos das ações apresentaram uma resposta positiva em relação à produção industrial e uma resposta negativa às mudanças nas taxas de juros reais; as mudanças na taxa de juros reais afetam negativamente a inflação e os retornos de ações; o retorno das ações respondeu com significância às mudanças no preço do petróleo
Maysami e Joh, 2000, Singapura, 1988-1995, VEC	1. Taxa de juros; 2. M2; 3. IPC; 4. IPI; 5. Taxa de juros interbancário	Os retornos das ações apresentaram uma resposta negativa às mudanças nas taxas de juros e positiva às alterações na taxa de câmbio; os fundamentos macroeconômicos dos EUA e do Japão afetam o preço das ações

	6. Outros	
Perales e Robins, 2002, México, 1990-2000, VAR e Causalidade de Granger	1. IPC; 2. Produção industrial; 3. M1; 4. Taxa de desemprego; 5. Taxa de juros; 6. taxa de câmbio; 7. Índice Dow Jones; 8. Taxa de juros dos títulos do tesouro norte americano.	H á uma grande influência do ciclo econômico norte americano no mercado financeiro mexicano
Pimenta Junior e Higuchi (2008)Brasil(1994-200, VAR e Causalidade de Granger	1. Taxa de juros; 2. Inflação; 3. Taxa de câmbio;	Nenhuma das variáveis macroeconômicas selecionadas apresentou uma relação de causalidade estatisticamente significativa em relação ao Ibovespa
Machado, Gartner e Machado (2017) Brasil (1999-2017) Markov-Switching	1.Ibovespa; 2.Inflação; 3.Taxa de juros; 4.M1; 5.PIB; 6. Importações; 7. Exportações; 8.Taxa de câmbi	As variáveis oferta de moeda, atividade econômica, nível de importação e exportação impactam negativamente o retorno de mercado, enquanto que as taxas de juros e de câmbio apresentaram um relacionamento positivo com este retorno
Bernardelli e Castro (2020) Brasil (2003-2019), GLS	1.Ibovespa; 2.Taxa de juros; 3.Receita do governo; 4.PIB; 5. Necessidade de financiamento do setor público; 7. Dow Jones; 8.Taxa de câmbio	Os resultados mostram que as variáveis macroeconômicas continuam exercendo influência sobre o Ibovespa, conformepreconiza a literatura, exceto, a proxy para a variável de estabilidade financeira do governo central
Fonte: Bernardelli e Castro (2020)		

Cada empresa pode ser afetada de maneira diferente por diferentes fatores, e é importante realizar uma análise cuidadosa em cada uma das ações e no próprio índice, visto que é possível

negociá-lo, acreditando no conjunto dessas empresas. Além disso, o desempenho do índice Bovespa é amplamente utilizado como um indicador da economia brasileira e do mercado financeiro brasileiro como um todo. O seu comportamento é acompanhado por investidores, analistas financeiros, empresas e pelo governo.

Diante deste cenário, este trabalho foi aprofundado em algumas variáveis independentes ao IBOV como será visto a seguir.

1.2. Impacto dos fatores macroeconômicos no índice BOVESPA

Fatores macroeconômicos são elementos que influenciam a economia de um país em nível geral. Eles incluem indicadores econômicos como a inflação, o PIB (Produto Interno Bruto), a taxa de juros, a taxa de câmbio, o nível de emprego, o consumo, entre outros. Esses fatores têm um impacto significativo nas decisões de investimento, no planejamento financeiro das empresas e das famílias, na política monetária e fiscal do governo e no desempenho do mercado financeiro. A análise desses fatores é importante para entender o comportamento da economia, prever tendências e tomar decisões estratégicas.

Um dos principais fatores macroeconômicos que pode influenciar a emissão de ações na bolsa de valores é a taxa de juros. Achsani & Strohe (2002) afirmam que a relação entre retornos acionários e taxa de juros são negativas pois um incremento na taxa implica em uma redução no preço das ações e consequentemente, redução no índice Ibov. Segundo Schor, a taxa de juros para o mercado brasileiro parte de duas hipóteses alternativas e com resultados contraditórios: primeiro, a alta taxa de juros inibe o retorno dos ativos ao mercado de ações; segundo, as empresas utilizam aplicações financeiras como fonte de lucro. A questão do efeito da taxa de juros no preço de ações para este autor não é clara e é necessário testar empiricamente.

Outro fator macroeconômico que pode afetar a decisão de uma empresa em emitir ações na bolsa de valores é o cenário político e econômico do país. Instabilidade política, incertezas econômicas e crises financeiras podem afetar a confiança dos investidores e tornar o mercado de ações menos atrativo para as empresas. Por outro lado, um cenário econômico estável e favorável pode incentivar a emissão de ações e atrair investidores.

1.3 Impacto da Selic, IPGM e IPCA no índice Bovespa

A taxa Selic é a taxa básica de juros da economia brasileira, determinada pelo Banco Central. Essa taxa afeta diretamente os custos de captação e investimento das empresas, e,

consequentemente, pode afetar o desempenho das ações negociadas na bolsa de valores. Quando a taxa Selic está alta, as empresas tendem a enfrentar maiores custos de captação e investimento, o que pode impactar negativamente o desempenho do mercado de ações. Por outro lado, quando a taxa Selic está baixa, as empresas podem encontrar oportunidades de financiamento mais acessíveis e investimentos mais atrativos, o que pode impulsionar o desempenho do mercado de ações.

O IGPM e o IPCA são índices de inflação que afetam os preços e a rentabilidade das empresas. Quando a inflação está alta, os preços dos insumos e serviços tendem a aumentar, o que pode afetar negativamente a rentabilidade das empresas e, consequentemente, o desempenho do mercado de ações. Por outro lado, quando a inflação está controlada, as empresas podem encontrar um ambiente mais favorável para o crescimento e expansão de suas atividades, o que pode impulsionar o desempenho do mercado de ações. Assim, para investidores e empresas, é importante compreender como essas variáveis afetam o mercado de ações e como elas podem influenciar a tomada de decisão de investimentos.

2. Problema Proposto

Qual é a magnitude do impacto desses indicadores no índice Bovespa? E como é possível utilizar técnicas de ciência de dados para prever o comportamento futuro do mercado de ações com base nesses indicadores macroeconômicos? Essas questões foram abordadas por meio de uma análise exploratória e modelagem estatística utilizando técnicas de regressão linear múltipla, análise de séries temporais, regressão regularizada Ridge, LASSO e arquitetura LSTM.

3. Objetivo

O objetivo é avaliar a magnitude do impacto desses indicadores no IBOV e como objetivos específicos, temos:

a. Verificar correlação entre os indicadores (IGPM, IPCA, Selic) e o índice Bovespa;

b. Identificar qual técnica possui melhor acerto na relação destes indicadores;

c. Criar modelos preditivos para o índice Bovespa utilizando Regressão Linear Múltipla e Ridge.

Os dados extraídos foram do período de 01/2010 a 04/2022. Para a análise exploratória, todos os dados foram considerados. Para os modelos preditivos, 70% dos dados foram utilizados como treinamento e 30% utilizado como teste.

4. Coleta de Dados

Foram utilizados três *datasets* base para a realização deste trabalho, sendo um referente ao valor de fechamento do índice Bovespa (dataset: ibov), outro referente à taxa Selic (dataset: selic) e outro unificado contemplando os indicadores IGPM e IPCA (dataset: igpm_ipca).

4.1 Tratamento do dataset ibov

Para a criação do dataset ibov, foi utilizada a biblioteca pandas-datareader e a API do Yahoo Finance (Yahoo, 2020). Após conectar com a API do Yahoo Finance, tratamos o dataset:

a. Importação das bibliotecas a serem utilizadas: pandas_datareader.data as web, yfinance como yf, pandas como pd, locale e parser:

```
import pandas_datareader.data as web
import yfinance as yf
import pandas as pd
import locale
from dateutil import parser
```

Sendo suas respectivas funções:

Biblioteca	Descrição
pandas_datareader.data	Biblioteca para coletar dados financeiros de diversas fontes.
yfinance	Biblioteca para coletar dados financeiros do Yahoo Finance.
pandas	Biblioteca de análise de dados que fornece estruturas de dados flexíveis e ferramentas de manipulação e análise de dados para Python.
locale	Biblioteca para a configuração de informações de localização, como formatação de datas, números e moedas, de acordo com o ambiente do usuário.
parser	Biblioteca para análise de datas e tempos em diversos formatos de entrada, incluindo ISO 8601 e formatos personalizados.

A seguir, o seguinte tratamento de dados foi realizado:

b. A localização UTF-8 foi configurada para que posteriormente seja possível identificar o formato da data e unificação com demais datasets;
c. Download dos dados de 01/2010 a 04/2023;
d. Reset do Index que é o campo Date;
e. Formatação do campo Date como MM/AAAA;
f. Remoção das colunas que não serão utilizadas neste estudo (Open, High, Low, Adj Close e Volume).
g. Definição da coluna 'Date' como o index do Dataframe;
h. Definição do último valor do mês como valor de fechamento do mês;

```
# Configurar o local para os Estados Unidos para o Yahoo Finance
locale.setlocale(locale.LC_ALL, 'en_US.UTF-8')

ibov = yf.download('^BVSP', start='2010-01-01', end='2023-04-01')
ibov = ibov.reset_index()

# Formatando a coluna 'data' para o formato MM/AAAA
ibov['Date'] = pd.to_datetime(ibov['Date'], format='%Y-%m-%d', dayfirst=True).dt.strftime('%d/%m/%Y')

ibov['Date'] = ibov['Date'].apply(lambda x: parser.parse(x).strftime('%d/%m/%Y'))

# Remove as colunas "Open", "High", "Low", "Adj Close" e "Volume"
ibov = ibov.drop(columns=["Open", "High", "Low", "Adj Close", "Volume", ])

# Converte a coluna 'Date' para o formato de data do Pandas
#ibov['Date'] = pd.to_datetime(ibov['Date'], format='%d/%m/%Y').dt.strftime('%m/%Y')
ibov['Date'] = pd.to_datetime(ibov['Date'], format='%d/%m/%Y')

# Define a coluna 'Date' como o índice do DataFrame
ibov = ibov.set_index('Date')

# Obtér o primeiro valor de cada mês
ibov = ibov.resample('MS').first()

# Converte o índice de volta para uma coluna
ibov = ibov.reset_index()

# Formata a coluna 'Date' para o formato MM/AAAA
ibov['Date'] = ibov['Date'].dt.strftime('%m/%Y')
```

Com o tratamento, finalizamos o dataset ibov, sendo que este possui as seguintes colunas:

Nome	Descrição	Tipo
Date	Data de fechamento – mensal	Index - Datetime
Close	Valor de fechamento do índice Bovespa	Float64

Os dados da tabela ibov foram utilizados nas análises descritiva, análise de séries temporais, regressão linear múltipla, análise de séries temporais, Ridge, LSTM, LASSO, sempre com base na coluna 'Close'.

4.2 Tratamento do dataset igpm_ipca

Os valores dos indicadores IGPM e IPCA foram obtidos através de repositórios do governo federal brasileiro, https://www3.bcb.gov.br/sgspub/localizarseries/localizarSeries.do?method=prepararTelaLocalizarSeries sendo os seguintes códigos utilizados na extração dos dados:

a. 189 - Índice geral de preços do mercado (IGP-M);
b. 433 - Índice nacional de preços ao consumidor-amplo (IPCA).

Foi realizado o upload do arquivo CSV no Colab e o tratamento dos dados:

a. Identificar a codificação do CSV padrão para “cp1252”;
b. Identificar o separador do CSV: “;”, visto que o separador padrão é “,”;
c. Identificar a casa decimal separada com “,” e não com “.”, como o padrão;
d. Substituir a “,” para “.” para termos o mesmo padrão de utilização do “,” e “.” que o dataset ibov;
e. Alterar o nome das colunas de: Data para Date e “189 - Índice geral de preços do mercado (IGP-M) - Var. % mensal” para “igpm” e “433 - Índice nacional de preços ao consumidor-amplo (IPCA) - Var. % mensal” para “ipca”;
f. Alterar o formato de leitura do campo date para MM/AAAA:

```
# Definir a função lambda para fazer a conversão de data
date_parser = lambda x: pd.datetime.strptime(x, '%d/%m/%Y')

# Ler o arquivo CSV
igpm_ipca = pd.read_csv('igpm_ipca.csv', encoding='cp1252', sep=';', parse_dates=['Data'], date_parser=date_parser)

igpm_ipca = igpm_ipca.replace(',', '.', regex=True)
igpm_ipca.columns = ['Date', 'igpm', 'ipca']
igpm_ipca['Date'] = pd.to_datetime(igpm_ipca['Date'], format='%d/%m/%Y')
igpm_ipca['Date'] = igpm_ipca['Date'].dt.strftime('%m/%Y')

print(igpm_ipca.columns)
print(igpm_ipca.dtypes)

print(igpm_ipca.head())
```

Com o tratamento, finalizamos o dataset igpm_ipca, sendo que este possui as seguintes colunas:

Nome	Descrição	Tipo
Date	Data de fechamento – mensal	Datetime
igpm	Valor do primeiro dia do mês do IGPM.	Float64

	Esse valor passa a ser considerado o valor mensal.	
ipca	Valor do primeiro dia do mês do IPCA. Esse valor passa a ser considerado o valor mensal.	Float64

Os dados do dataframe igpm_ipca foram utilizados na regressão linear múltipla, Ridge e LSTM, com utilização das colunas ipgm e ipca.

4.3 Tratamento do dataset selic

Os valores da Selic foram obtidos através de repositórios do governo federal brasileiro, https://www3.bcb.gov.br/sgspub/localizarseries/localizarSeries.do?method=prepararTelaLocalizarSeries sendo os seguintes códigos: 432 Meta Selic definida pelo Copom.

Foi realizado o upload do arquivo CSV no Colab e o tratamento dos dados:

g. Identificar o separador do CSV: “;”, visto que o separador padrão é “,”;
h. Identificar a casa decimal separada com “,” e não com “.”, como o padrão;
i. Substituir a “,” para “.” para termos o mesmo padrão de utilização do “,” e “.” que o dataset ibov;
j. Alterar o nome das colunas de: Data para Date e 432 - Taxa de juros - Meta Selic definida pelo Copom - % a.a. para “selic”;
k. Alterar o formato de leitura do campo date para MM/AAAA:

```
# Ler o arquivo CSV
selic = pd.read_csv('STP-selic.csv', sep=';', decimal=',')

selic = selic.replace(',', '.', regex=True)

# Definir os nomes das colunas
selic.columns = ['Date', 'selic']

# Converte a coluna 'Date' para o formato datetime
selic['Date'] = pd.to_datetime(selic['Date'])

# Selecionando apenas as linhas em que o dia é igual a 1
selic = selic.loc[selic['Date'].dt.day == 1]

# Formatando a coluna 'data' para o formato MM/AAAA
selic['Date'] = selic['Date'].dt.strftime('%m/%Y')
```

Com o tratamento, finalizamos o dataset selic, sendo que este possui as seguintes colunas:

Nome	**Descrição**	**Tipo**

Date	Data de fechamento – mensal	Datetime
selic	Valor do primeiro dia do mês do Selic. Esse valor passa a ser considerado o valor mensal.	Float64

Os dados do dataframe selic foram utilizados na regressão linear múltipla e Ridge, com utilização da coluna selic.

4.4 Unificação dos dados em tabela única

Após realizar o upload e conexão com os indicadores desejados, foi criada um dataset único, nomeado de tabelaunificada, contendo todos os indicadores e com merge realizado sobre a data:

```
[179] # Merge entre as tabelas ibov e igpm_ipca
      tabelaunificada = pd.merge(igpm_ipca, selic, on='Date', how='left')

      # Merge entre as tabelas df e selic
      tabelaunificada = pd.merge(tabelaunificada, ibov, on='Date', how='left')

      # Transformar a coluna 'Date' em um tipo de dado datetime
      tabelaunificada['Date'] = pd.to_datetime(tabelaunificada12['Date'])

      print(tabelaunificada)
      print(tabelaunificada.dtypes)
```

```
      print(tabelaunificada.head())
```

```
        Date    Close  selic  igpm  ipca
0 2010-01-01  66572.0   8.75  0.63  0.75
1 2010-02-01  67163.0   8.75  1.18  0.78
2 2010-03-01  67109.0   8.75  0.94  0.52
3 2010-04-01  70045.0   8.75  0.77  0.57
4 2011-01-01  67847.0  10.75  0.79  0.83
```

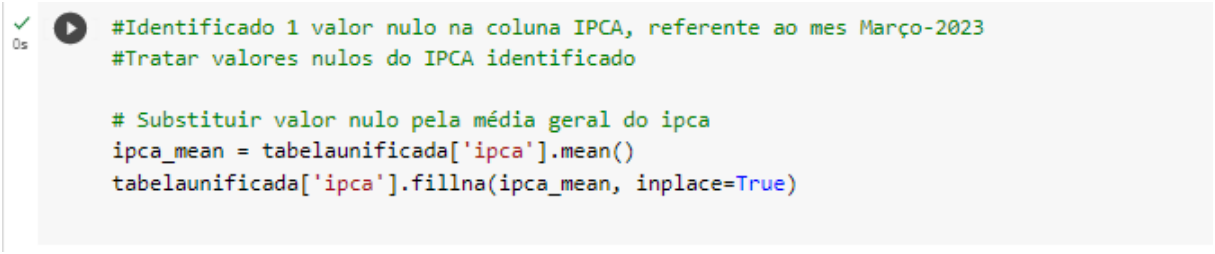
Este dataset será o utilizado nas análises de regressão linear multipla e contem os seguintes campos:

Nome	**Descrição**	**Tipo**
Date	Data de fechamento – mensal	Datetime
Close	Valor do mês referente ao fechamento do ibov	Float64
selic	Valor do primeiro dia do mês do Selic.	Float64

	Esse valor passa a ser considerado o valor mensal.	
igpm	Valor do primeiro dia do mês do IGPM. Esse valor passa a ser considerado o valor mensal.	Float64
ipca	Valor do primeiro dia do mês do IPCA. Esse valor passa a ser considerado o valor mensal.	Float64

4.5 Tratamento da tabelaunificada

Após a criação da tabela, foi necessário avaliar necessidade de tratamento de dados. Para tal, foi identificado se existiam valores nulos ou duplicados e foi identificado um valor nulo na coluna IPCA. Para correção, o mesmo foi substituído pela média do indicador e novamente foi avaliado a existência de valores nulos ou duplicados:



```
#Identificado 1 valor nulo na coluna IPCA, referente ao mes Março-2023
#Tratar valores nulos do IPCA identificado

# Substituir valor nulo pela média geral do ipca
ipca_mean = tabelaunificada['ipca'].mean()
tabelaunificada['ipca'].fillna(ipca_mean, inplace=True)
```

```python
# contar o número de linhas duplicadas
num_duplicatas = tabelaunificada.duplicated().sum()

print(f'O DataFrame Tabela Unificada tem {num_duplicatas} linhas duplicadas.')

# contar se existem valores nulos para tratamento

nulos1 = tabelaunificada['Close'].isnull().sum()
nulos2 = tabelaunificada['selic'].isnull().sum()
nulos3 = tabelaunificada['igpm'].isnull().sum()
nulos4 = tabelaunificada['ipca'].isnull().sum()

print(f"A coluna Close tem {nulos1} valores nulos.")
print(f"A coluna2 selic {nulos2} valores nulos.")
print(f"A coluna3 igpm {nulos3} valores nulos.")
print(f"A coluna4 ipca {nulos4} valores nulos.")

print(tabelaunificada.dtypes)
print(tabelaunificada)
```

```
O DataFrame Tabela Unificada tem 0 linhas duplicadas.
A coluna Close tem 0 valores nulos.
A coluna2 selic 0 valores nulos.
A coluna3 igpm 0 valores nulos.
A coluna4 ipca 0 valores nulos.
Date     datetime64[ns]
igpm            float64
ipca            float64
selic           float64
Close           float64
```

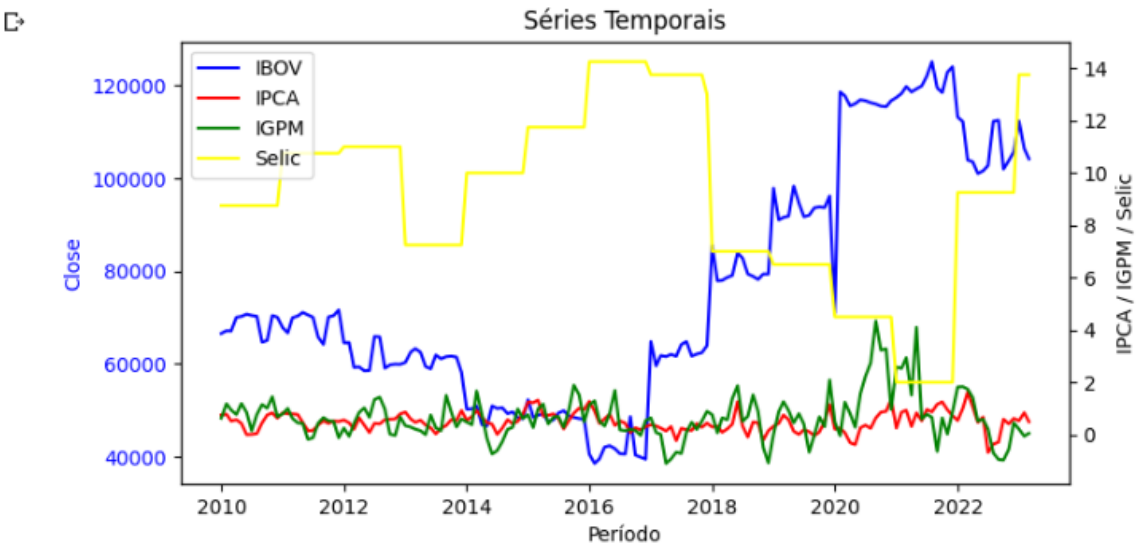
Ainda, para fins de utilização nos modelos de Machine Learning, foi criado um dataset com os dados normalizados:

```python
from sklearn.preprocessing import MinMaxScaler

# Salvar a coluna 'Date' em uma variável separada
coluna_data = tabelaunificada['Date']

# Selecionar apenas as colunas numéricas
dados_numericos = tabelaunificada.select_dtypes(include='number')

# Normalizar a tabela
scaler = MinMaxScaler()
tabela_normalizada = pd.DataFrame(scaler.fit_transform(dados_numericos), columns=dados_numericos.columns)

# Concatenar a coluna 'Date' com a tabela normalizada
tabela_normalizada = pd.concat([coluna_data, tabela_normalizada], axis=1)

print(tabela_normalizada.head())
print(tabela_normalizada.dtypes)
```

A partir da tabela normalizada, foi possível visualizar um gráfico correlacionando os indicadores:

```python
import matplotlib.pyplot as plt

# Criando os eixos y
fig, ax1 = plt.subplots(figsize=(8, 4))

# Plotando a série ibov no primeiro eixo y
ax1.plot(tabelaunificada['Date'], tabelaunificada['Close'], color='blue', label='IBOV')
ax1.set_xlabel('Período')
ax1.set_ylabel('Close', color='blue')
ax1.tick_params(axis='y', labelcolor='blue')

# Criando o segundo eixo y
ax2 = ax1.twinx()

# Plotando as séries ipca e igpm no segundo eixo y
ax2.plot(tabelaunificada['Date'], tabelaunificada['ipca'], color='red', label='IPCA')
ax2.plot(tabelaunificada['Date'], tabelaunificada['igpm'], color='green', label='IGPM')
ax2.plot(tabelaunificada['Date'], tabelaunificada['selic'], color='yellow', label='Selic')

ax2.set_ylabel('IPCA / IGPM / Selic', color='black')
ax2.tick_params(axis='y', labelcolor='black')

# Adicionando uma legenda
lines, labels = ax1.get_legend_handles_labels()
lines2, labels2 = ax2.get_legend_handles_labels()
ax2.legend(lines + lines2, labels + labels2, loc='upper left')

plt.title('Séries Temporais')
plt.show()
```

Tendo como resultado o seguinte gráfico para análise das correlações:



5. Análise e Exploração do IBOV

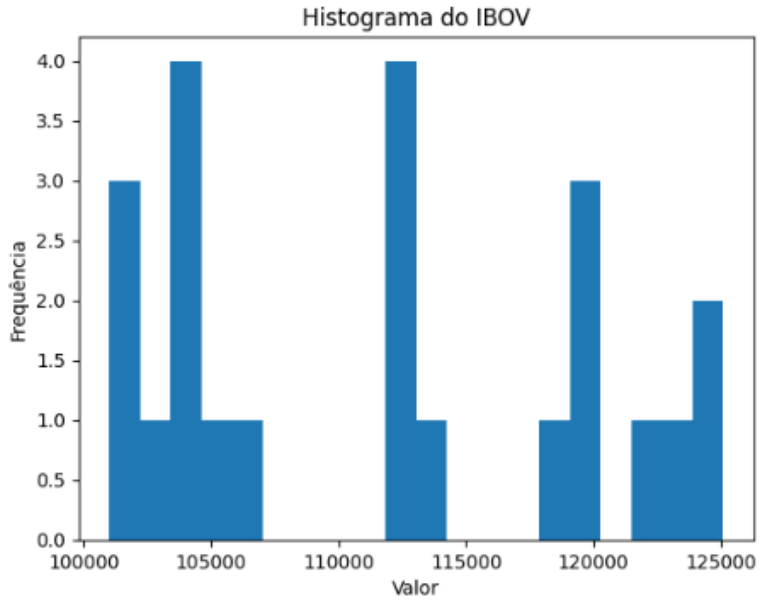
Foi iniciada a análise e exploração do índice Bovespa (ibov) com um resumo estatístico. Para a análise descritiva, foi considerado apenas os últimos 24 meses e para isso, ocorreu a criação do dataset ibovdescribe:

```
[11] from pandas.core.describe import describe_ndframe
     from datetime import datetime, timedelta

     # Filtra para manter apenas as linhas com datas a partir de 24 meses atrás
     data_limite = datetime.now() - timedelta(days=24*30)
     tabela_filtrada = tabelaunificada.loc[tabelaunificada['Date'] >= data_limite]
```

Na sequência, foi analisado dados estatísticos da série, consierando apenas os últimos 24 meses:

```
# Obter as estatísticas descritivas da coluna 'Close'
ibovdescribe = tabela_filtrada['Close'].describe()

ibovdescribe.describe()
print("Valores mínimos e máximos:")
print("IBOV: mínimo = {:.2f} ({}) , máximo = {:.2f} ({})".format(tabela_filtrada['Close'].min(),
                                                                 tabela_filtrada.loc[tabela_filtrada['Close'].idxmin(),'Date'],
                                                                 tabela_filtrada['Close'].max(),
                                                                 tabela_filtrada.loc[tabela_filtrada['Close'].idxmax(),'Date']))

# Histograma do IBOV
plt.hist(tabela_filtrada['Close'], bins=20)
plt.title('Histograma do IBOV')
plt.xlabel('Valor')
plt.ylabel('Frequência')
plt.show()

print(ibovdescribe)
```

```
Valores mínimos e máximos:
IBOV: mínimo = 101006.00 (2022-05-01 00:00:00) , máximo = 125077.00 (2021-08-01 00:00:00)
```

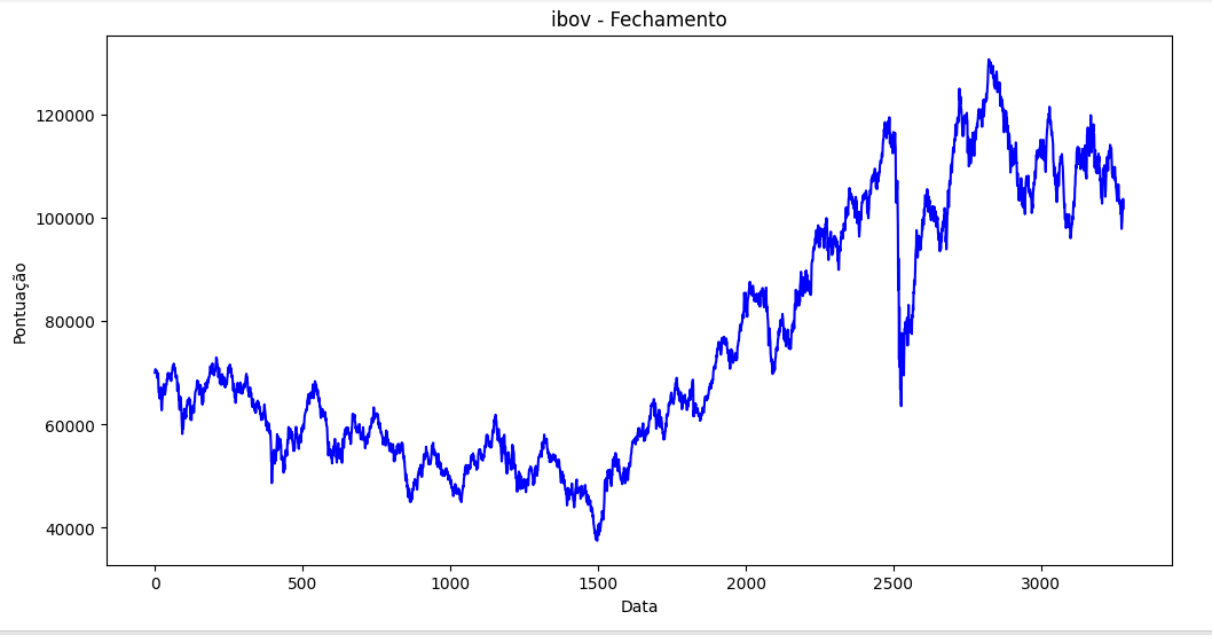



```
count        23.000000
mean     111641.347826
std        8242.791548
min      101006.000000
25%      103850.500000
50%      112234.000000
75%      119326.000000
max      125077.000000
Name: Close, dtype: float64
```

Com isso foi identificado que a mínima no período citado ocorreu em Maio/2022 e máxima ocorreu em Agosto/2021.

Para aprofundar em nossa análise, foi criada uma série temporal:

```
# Plotando gráfico de linha para visualização da série temporal
plt.figure(figsize=(12,6))
sns.lineplot(data=ibov['Close'], color='blue')
plt.title('ibov - Fechamento')
plt.xlabel('Data')
plt.ylabel('Pontuação')
plt.show()
```



Após a visualização da série, foi aplicado o Teste Dickey-Fuller para identificar se a série é estacionária ou não estacionária. Para tal, o dataset ibov, coluna Close foi avaliado, considerando o período 01/2010 a 04/2023:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from statsmodels.tsa.stattools import adfuller

# Teste de Dickey-Fuller aumentado para verificar estacionariedade
resultado = adfuller(ibov['Close'])
print('Estatística ADF:', resultado[0])
print('Valor p:', resultado[1])
print('Valores críticos:')
for chave, valor in resultado[4].items():
    print(f'   {chave}: {valor}')
if resultado[0] < resultado[4]['5%']:
    print('A série é estacionária')
else:
    print('A série não é estacionária')
```

```
Estatística ADF: -0.7213039841626514
Valor p: 0.8411538879691998
Valores críticos:
   1%: -3.6055648906249997
   5%: -2.937069375
   10%: -2.606985625
A série não é estacionária
```

Como resultado, a estatística ADF é de -0.32 e o valor p é de 0.92. Como o valor-p é maior que o nível de significância comumente escolhido de 0.05, não podemos rejeitar a hipótese nula de que a série é não estacionária.

6. Análise de Séries Temporais

Para realizar a predição do ibov através de séries temporais, os pacotes **upgrade statsmodels pmdarima** e **upgrade pmdarima**. Após a instalação, as bibliotecas **numpy**,**ARIMA** e **sklearn.metrics** foram importadas:

```
[21] !pip install --upgrade statsmodels
     !pip install pmdarima
     !pip install --upgrade pmdarima
```

```
import numpy as np
from statsmodels.tsa.arima.model import ARIMA
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_squared_error
```

Para análise exclusiva de série temporária, o ***dataset* ibov2** foi criado, derivado do ***dataset* tabelaunificada**, porém mantida apenas a coluna **Close** e o dataset foi dividido em teste e treinamento:

```
#Criacao de dataframe para analise de serie nao estacionaria
ibov2 = tabelaunificada.drop(columns=["igpm", "selic", "ipca"])

# Separando a base de treinamento e teste (70% e 30%, respectivamente)
train_size = int(len(ibov2)*0.7)
train_set = ibov2[:train_size]
test_set = ibov2[train_size:]

# Verificando o tamanho dos conjuntos de treinamento e teste
print(len(train_set), len(test_set))
```

Dado que a série foi classificada como não estacionária, foi selecionado o método ARIMA e este modelo não foi capaz de nos dar um melhor modelo, utilizando os dados selecionados:

```
#Criacao de dataframe para analise de serie nao estacionaria
ibov2 = tabelaunificada.drop(columns=["igpm", "selic", "ipca"])

# Separando a base de treinamento e teste (70% e 30%, respectivamente)
train_size = int(len(ibov2)*0.7)
train_set = ibov2[:train_size]
test_set = ibov2[train_size:]

# Verificando o tamanho dos conjuntos de treinamento e teste
print(len(train_set), len(test_set))

# Cria uma lista com todas as possíveis combinações de p, d, q
p = d = q = range(0, 4)
pdq = list(itertools.product(p, d, q))

# Testa todas as combinações de pdq e seleciona o modelo com o menor AIC
aic_values = []
for param in pdq:
    try:
        model = sm.tsa.ARIMA(ibov2["Close"], order=param)
        results = model.fit()
        aic_values.append((param, results.aic))
    except:
        continue

if aic_values:
    best_model = min(aic_values, key=lambda x: x[1])
    print("Best ARIMA model:", best_model[0], " AIC:", best_model[1])
else:
    print("Nenhum modelo ARIMA possível encontrado")
```

```
111 48
Nenhum modelo ARIMA possível encontrado
```

Assim, o método ARIMA foi descontinuado dado o não ajuste dos dados ao modelo.

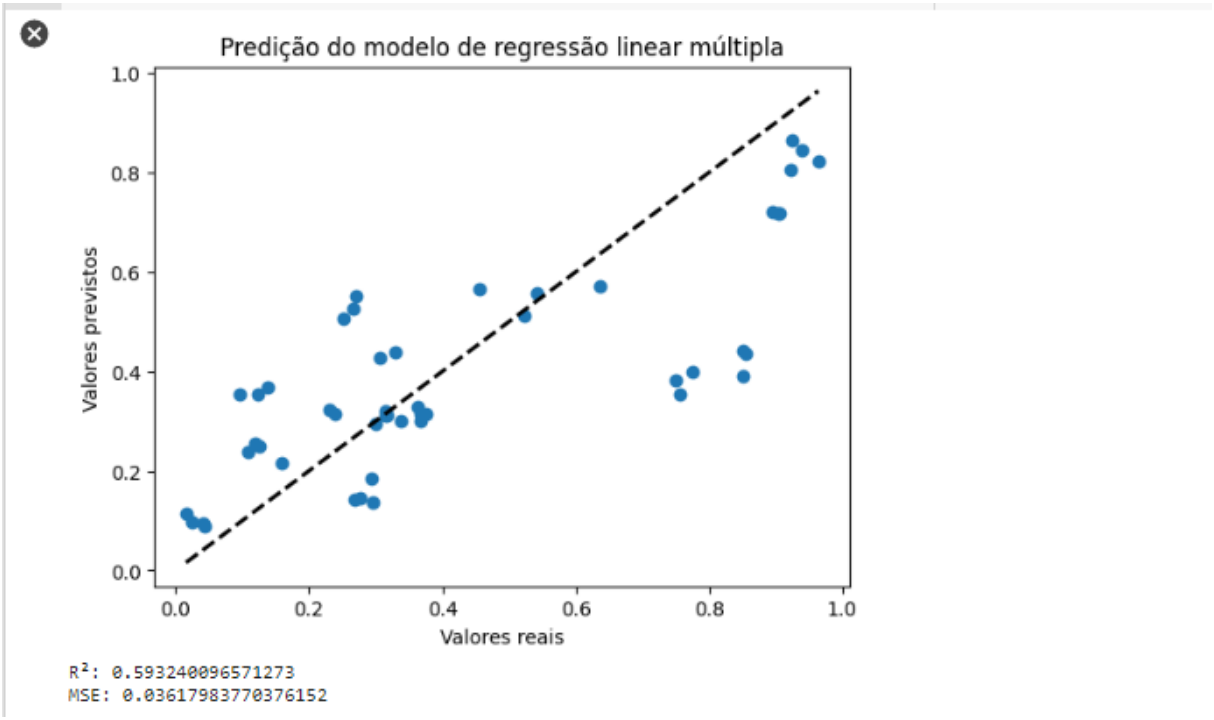
7. Regressão Linear Múltipla

Foi aplicada a regressão linear múltipla dado a natureza do problema, para analisar a relação entre a variável dependente (ibov) com as variáveis independentes (selic, igpm, ipca). No contexto do presente trabalho os indicadores podem ser considerados independentes e

influenciar o Ibov. Consequentemente, entender a relação dessas variáveis pode afetar a decisão do investidor. Para essa análise, foi as bibliotecas sklearn de regressão linear foi importada:

```
# Definir as variáveis independentes (X) e a variável dependente (y)
X = tabela_normalizada[['selic', 'igpm', 'ipca']]
y = tabela_normalizada['Close']

# Dividir os dados em conjunto de treinamento e conjunto de teste
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Treinar o modelo de regressão linear múltipla
modelo = LinearRegression()
modelo.fit(X_train, y_train)

# Fazer as previsões do modelo nos dados de teste
y_pred = modelo.predict(X_test)

# Visualizar a predição
plt.scatter(y_test, y_pred)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=2) # linha da previsão perfeita
plt.xlabel('Valores reais')
plt.ylabel('Valores previstos')
plt.title('Predição do modelo de regressão linear múltipla')
plt.show()

# Calcular o R²
r2 = r2_score(y_test, y_pred)

# Calcular o MSE
mse = mean_squared_error(y_test, y_pred)
```

No nosso caso, o R^2 obtido foi de 0.59, o que indica que as variáveis independentes explicam cerca de 59% da variância na coluna 'Close', o que é um ajuste médio. A partir daqui e levando em consideração o referencial teórico, avalia-se que há a necessidade de incluir outras variáveis para que o modelo explique melhor a variável dependente:



7. Criação de modelos de machine learning

Neste tópico será descrito os modelos preditivos do ibov, criados em Python, utilizando a biblioteca ScikitLearn e arquitetura LSTM no Keras.

7.1 Ridge Regression

Para essa análise, será utilizada a tabela que já está unificada e normalizada descrita no tópico “4.5 Tratamento de Dados”. Após definição do dataframe a ser utilizado, as bases treinamento e teste, respeitando a proporção 70% e 30% foram definidas:

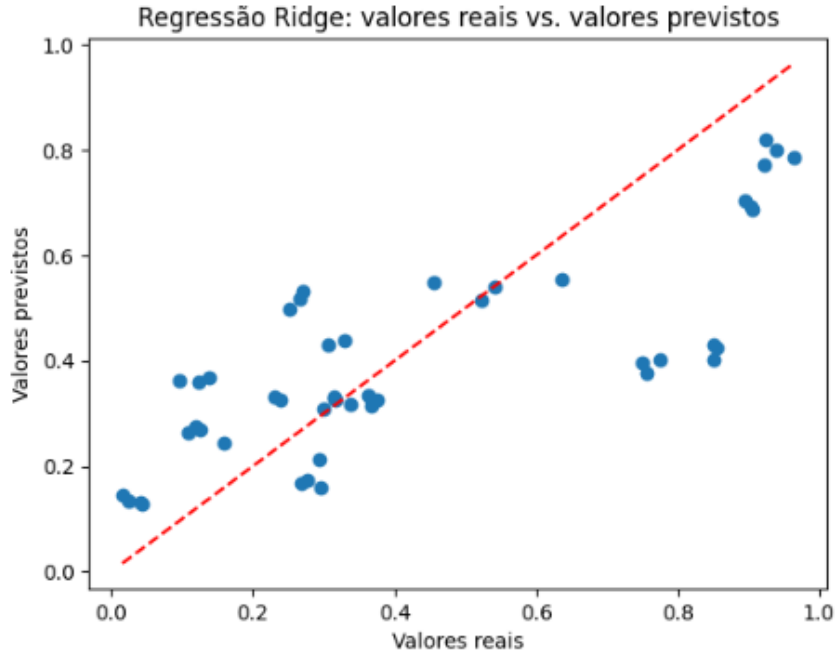
```
from sklearn.model_selection import train_test_split

X = tabela_normalizada[['ipca', 'igpm', 'selic',]]  # variáveis independentes
y = tabela_normalizada['Close']  # variável dependente

# divide em 70% para treinamento e 30% para teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

O modelo então será criado e treinado. A classe Ridge foi importada do pacote sklearn.linear_model e o objeto ridge_model foi criado especificando o Ridge com o parâmetro = 1; o método fit foi aplicado passando os dados de treino como parâmetro. O r^2 medido foi de 0.57 e o MSE foi 0.03. Com a regressão Ridge, houve um ajuste aos dados, e explica 57% da variação na variável dependente, o mais alto e similar a regressão linear múltipla:

```
[26] from sklearn.linear_model import LinearRegression
     from sklearn.linear_model import Ridge
     import matplotlib.pyplot as plt

     # define o modelo
     model = LinearRegression()

     # Cria o modelo Ridge
     ridge_model = Ridge(alpha=1)

     # Treina o modelo
     ridge_model.fit(X_train, y_train)

     # Faz a previsão dos valores de y para os dados de teste
     y_pred_ridge = ridge_model.predict(X_test)

     # Calcula o R2 do modelo Ridge
     r2_ridge = ridge_model.score(X_test, y_test)
     print("R2 do modelo Ridge:", r2_ridge)

     # Calcula o MSE do modelo Ridge
     mse_ridge = mean_squared_error(y_test, y_pred_ridge)
     print("MSE do modelo Ridge:", mse_ridge)

     # plota o gráfico com os resultados
     plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], '--', color='red')
     plt.scatter(y_test, y_pred_ridge)
     plt.xlabel('Valores reais')
     plt.ylabel('Valores previstos')
     plt.title('Regressão Ridge: valores reais vs. valores previstos')
     plt.show()
```

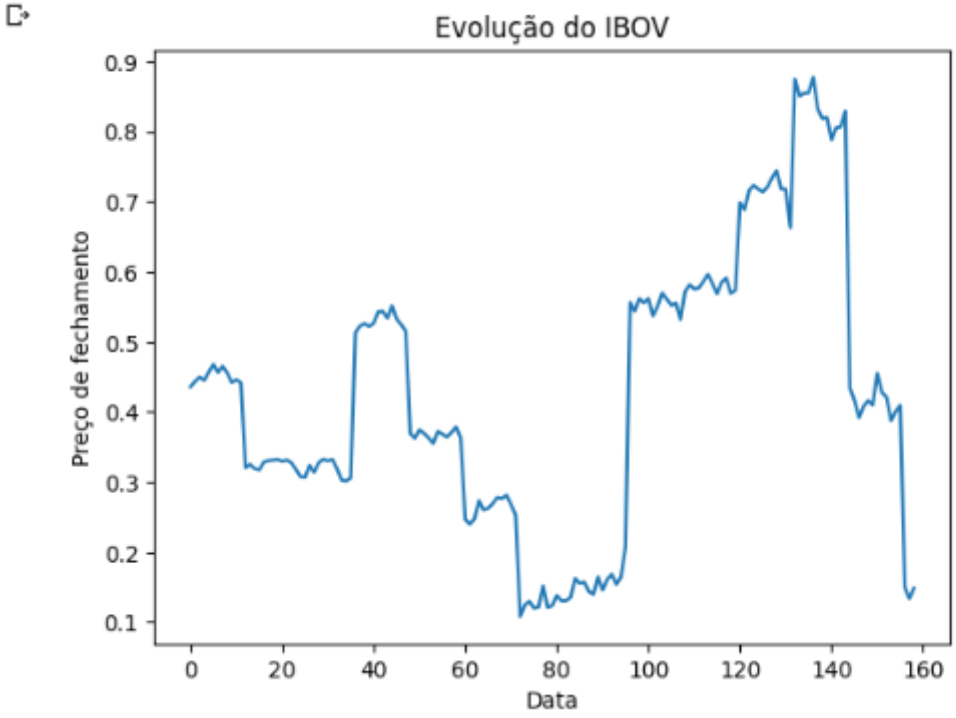
O seguinte gráfico foi plotado a fim de analisar o modelo:

```
R2 do modelo Ridge: 0.5793416987062903
MSE do modelo Ridge: 0.03741605045447451
```



Por fim, foi criado um novo dataset ibov3, aplicando a regressão ridge em sua totalidade:

```
# Treinar o modelo Ridge usando todos os dados da tabelaunificada
ridge_model_all = Ridge(alpha=1)
ridge_model_all.fit(X, y)

# Fazer previsões para todos os dados da tabelaunificada
y_pred_all = ridge_model_all.predict(X)

# Adicionar as previsões ao DataFrame ibov3
ibov3 = pd.DataFrame({'Close': y_pred_all}, index=tabelaunificada.index)

#Gráfico de linha com a evolução do Ibovespa ao longo do tempo:
plt.plot(ibov3["Close"])
plt.title("Evolução do IBOV")
plt.xlabel("Data")
plt.ylabel("Preço de fechamento")
plt.show()
```



7.2 Arquitetura LSTM

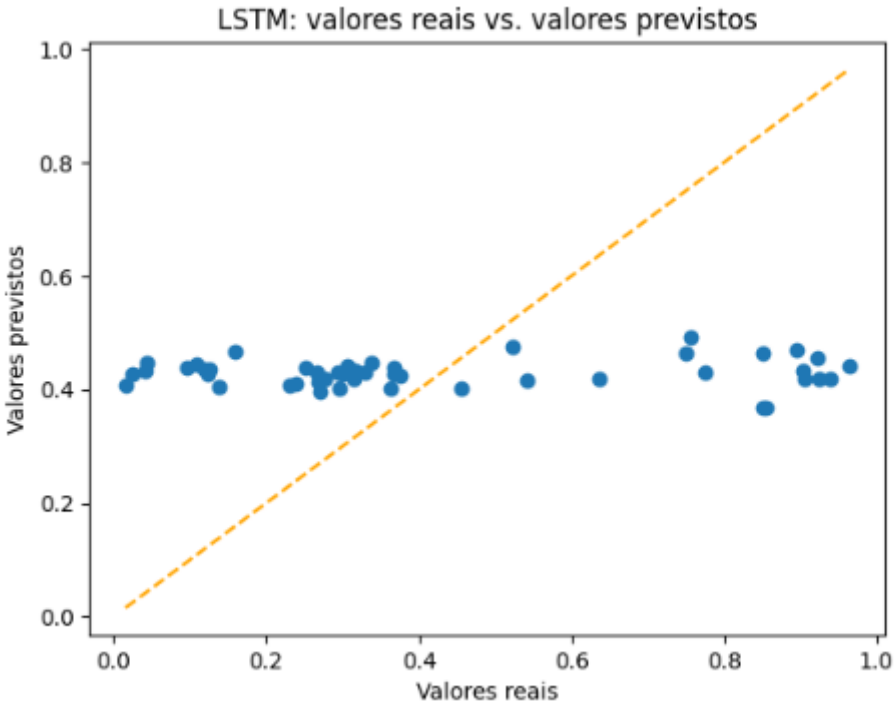
Para apresentar os resultados do modelo, inicialmente foram importadas as bibliotecas do Keras:

```
import numpy as np
import pandas as pd
from keras.models import Sequential
from keras.layers import Dense, LSTM
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, r2_score
```

Em seguida, houve a definição do modelo, compilamento e treino:

```
# definir o modelo
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(X_train.shape[1], 1)))
model.add(LSTM(units=50))
model.add(Dense(1))

# compilar o modelo
model.compile(loss='mean_squared_error', optimizer='adam')

# treinar o modelo
model.fit(X_train, y_train, epochs=30, batch_size=32)

# avaliar o modelo
test_predict = model.predict(X_test)

# prever valores para os dados de teste
y_pred = model.predict(X_test)

# calcular o r2
r2 = r2_score(y_test, y_pred)
print(f"R2: {r2:.4f}")

# calcular o MSE
mse = mean_squared_error(y_test, y_pred)
print(f"MSE: {mse:.4f}")
```

Após o treinamento, a predição, r^2 e MSE. O R2 foi de 0.03 e o MSE 0.08, sendo este modelo, o que apresentou menor aproveitamento das variaveis independentes com a variável dependente:



7.3 Regressão LASSO

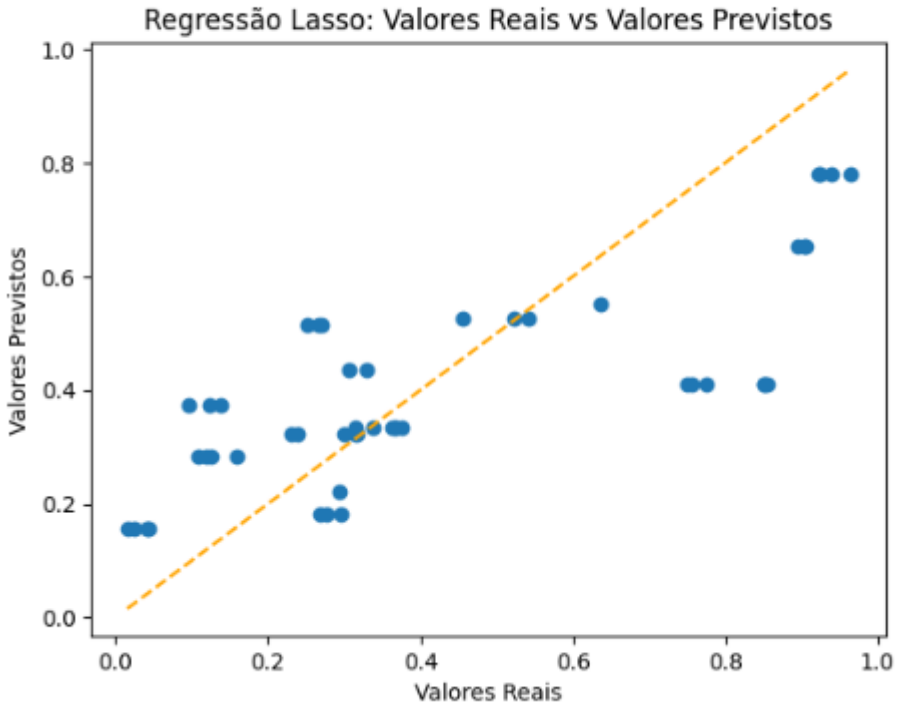
A fins de efeito comparativo e identificação do melhor modelo, foi gerado um modelo de regressão tipo LASSO. Foi utilizada a biblioteca sklearn.linear_model e um alpha de =0.01 foi definido:

```
[ ]  from sklearn.linear_model import Lasso
     from sklearn.model_selection import train_test_split
     from sklearn.metrics import mean_squared_error
```

```
from sklearn.linear_model import Lasso
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

# divide em 70% para treinamento e 30% para teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# cria um objeto Lasso com um alpha de 0.01
lasso = Lasso(alpha=0.01)

# treina o modelo com os dados de treinamento
lasso.fit(X_train, y_train)

# faz as previsões com o conjunto de teste
y_pred = lasso.predict(X_test)

# calcula o erro quadrático médio das previsões
mse = mean_squared_error(y_test, y_pred)
print("Erro quadrático médio:", mse)

# verifica os coeficientes das variáveis independentes
print("Coeficientes:", lasso.coef_)

# y_true são os valores reais da variável dependente
# y_pred são as previsões feitas pelo modelo Lasso
r2 = r2_score(y_test, y_pred)
print("R²:", r2)
```

Como resultado obtido, houve o MSE de 0.56 e r² de 0.03. Assim LASSO manteve resultados similares ao demais modelos:

```
Erro quadrático médio: 0.039006977502159194
Coeficientes: [-0.         0.        -0.6239427]
R²: 0.5614553461588577
```



8. Conclusão

As variáveis independentes do ibov, o IGPM, IPCA e Selic apresentam nesse estudo média correlação com o modelo e podem explicar até 59% da variável Ibov. Além disso, foi possível identificar que a série é não estacionária e pode sofrer variação devido a sazonalidades, instabilidades políticas ou alterações no plano econômico. Outro ponto é que o ibov é o conjunto do resultado de mais de 70 ações e essas, individualmente podem sofrer alterações e fazer que o índice oscile.

Por fim, identificou-se que através da Regressão Linear Multipla que os indicadores IGPM, IPCA e Selic explicam 59% da variação do ibov, sendo necessário adicionar mais variáveis para um modelo mais assertivo.

9. Links

Aqui você deve disponibilizar os links para o vídeo com sua apresentação de 5 minutos e para o repositório contendo os dados utilizados no projeto, scripts criados, etc.

Link para o vídeo: https://youtu.be/FqFX5sokbJU
Link para o repositório: https://github.com/yumids/tcc_pucminas.git

Referências

ACHSANI, N.A.; STROHE, H.G. **Stock market returns and macroeconomic factors: evidence from Jakarta stock exchange of Indonesia** 1990-2001. Extraído em 04 de fevereiro de https://scholar.google.com/citations?user=Cim_ngoAAAAJ&hl=en

GRÔPPO, Gustavo de Souza. **Causalidade das variáveis macroeconômicas sobre o IBOVESPA.** Piracicaba: Digital Libraby USP, 2004. Extraído em 01 de março de 2023, https://www.teses.usp.br/teses/disponiveis/11/11132/tde-06012005-165535/publico/gustavo.pdf

Bernardelli, L. V., & de Castro, G. H. L. (2020). **Mercado acionário e variáveis macroeconômicas: evidências para o Brasil**. Revista Catarinense de Ciência Contábil, 19, 2892.

Bernardelli, Luan V.; Borges, Murilo J.; Sanches, Simone L. R.; Sbardellati, Elaine C. de A. **A relação entre as variáveis macroeconômicas e o Ibovespa: Novas evidências para o Brasil**. Revista Mineira De Contabilidade, 21(3), 97–112. https://doi.org/10.51320/rmc.v21i3.1164

B3, Brasil, Bolsa e Balcão. (2019). Índice Bovespa (Ibovespa) Março 13, 2023, from https://www.b3.com.br/pt_br/market-data-e-indices/indices/indices-amplos/ibovespa.htm#:~:text=O%20Ibovespa%20%C3%A9%20o%20principal,investidores%20ao%20redor%20do%20mundo.

MUKHERJEE, T.; NAKA, A. Dynamic relations between macroeconomic variables and Japanese stock Market: an application of a vector error correction model. The Journal of Financial Research, v18, n.2, p.22-237, Summer 1995. Extraído em 3 de março de 2023
Link: https://onlinelibrary.wiley.com/doi/10.1111/j.1475-6803.1995.tb00563.x

Apêndice

Script Completo

```
import pandas as pd
from google.colab import files

!pip install pandas-datareader
!pip install yfinance
!pip install matplotlib
import pandas_datareader.data as web
import yfinance as yf
import pandas as pd
import locale
from dateutil import parser

locale.setlocale(locale.LC_ALL, 'en_US.UTF-8')
ibov = yf.download('^BVSP', start='2010-01-01', end='2023-04-01')
ibov = ibov.reset_index()
ibov['Date'] = pd.to_datetime(ibov['Date'], format='%Y-%m-
%d', dayfirst=True).dt.strftime('%d/%m/%Y')
ibov['Date'] = ibov['Date'].apply(lambda x: parser.parse(x).strftime('%
d/%m/%Y'))
ibov = ibov.drop(columns=["Open", "High", "Low", "Adj Close", "Volume",
 ])
ibov['Date'] = pd.to_datetime(ibov['Date'], format='%d/%m/%Y')
ibov = ibov.set_index('Date')
ibov = ibov.resample('MS').first()
ibov = ibov.reset_index()
ibov['Date'] = ibov['Date'].dt.strftime('%m/%Y')

date_parser = lambda x: pd.datetime.strptime(x, '%d/%m/%Y')
locale.setlocale(locale.LC_ALL, 'en_US.UTF-8')
igpm_ipca = pd.read_csv('igpm_ipca.csv', encoding='cp1252', sep=';', pa
rse_dates=['Data'], date_parser=date_parser)
igpm_ipca = igpm_ipca.replace(',', '.', regex=True)
igpm_ipca.columns = ['Date', 'igpm', 'ipca']
igpm_ipca['Date'] = pd.to_datetime(igpm_ipca['Date'], format='%d/%m/%Y'
)
igpm_ipca['Date'] = igpm_ipca['Date'].dt.strftime('%m/%Y')
igpm_ipca['igpm'] = pd.to_numeric(igpm_ipca['igpm'], errors='coerce')
igpm_ipca['ipca'] = pd.to_numeric(igpm_ipca['ipca'], errors='coerce')

selic = pd.read_csv('STP-selic.csv', sep=';', decimal=',')
selic = selic.replace(',', '.', regex=True)
selic.columns = ['Date', 'selic']
selic['Date'] = pd.to_datetime(selic['Date'])
```

```
selic = selic.loc[selic['Date'].dt.day == 1]
selic['Date'] = selic['Date'].dt.strftime('%m/%Y')

tabelaunificada = pd.merge(igpm_ipca, selic, on='Date', how='left')
tabelaunifica-
da = pd.merge(tabelaunificada, ibov, on='Date', how='left')
tabelaunificada['Date'] = pd.to_datetime(tabelaunificada12['Date'])
ipca_mean = tabelaunificada['ipca'].mean()
tabelaunificada['ipca'].fillna(ipca_mean, inplace=True)
num_duplicatas = tabelaunificada.duplicated().sum()

from sklearn.preprocessing import MinMaxScaler
coluna_data = tabelaunificada['Date']
dados_numericos = tabelaunificada.select_dtypes(include='number')
scaler = MinMaxScaler()
tabe-
la_normalizada = pd.DataFrame(scaler.fit_transform(dados_numericos), co
lumns=dados_numericos.columns)
tabe-
la_normalizada = pd.concat([coluna_data, tabela_normalizada], axis=1)
from pandas.core.describe import describe_ndframe
from datetime import datetime, timedelta
data_limite = datetime.now() - timedelta(days=24*30)
tabe-
la_filtrada = tabelaunificada.loc[tabelaunificada['Date'] >= data_limit
e]
ibovdescribe = tabela_filtrada['Close'].describe()
ibovdescribe.describe()
print("Valores mínimos e máximos:")
print("IBOV: mínimo = {:.2f} ({}) , máximo = {:.2f} ({})".format(tabela
_filtrada['Close'].min(),
                                                                tabe-
la_filtrada.loc[tabela_filtrada['Close'].idxmin(),'Date'],
                                                                tabe-
la_filtrada['Close'].max(),
                                                                tabe-
la_filtrada.loc[tabela_filtrada['Close'].idxmax(),'Date']))
plt.hist(tabela_filtrada['Close'], bins=20)
plt.title('Histograma do IBOV')
plt.xlabel('Valor')
plt.ylabel('Frequência')
plt.show()

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error
```

```
X = tabela_normalizada[['selic', 'igpm', 'ipca']]
y = tabela_normalizada['Close']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3
, random_state=42)
modelo = LinearRegression()
modelo.fit(X_train, y_train)
y_pred = modelo.predict(X_test)
plt.scatter(y_test, y_pred)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k
--', lw=2)
plt.xlabel('Valores reais')
plt.ylabel('Valores previstos')
plt.title('Predição do modelo de regressão linear múltipla')
plt.show()
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
print('R²:', r2)
print('MSE:', mse)

!pip install statsmodels
!pip install seaborn
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from statsmodels.tsa.stattools import adfuller
resultado = adfuller(ibov['Close'])
print('Estatística ADF:', resultado[0])
print('Valor p:', resultado[1])
print('Valores críticos:')
for chave, valor in resultado[4].items():
    print(f'   {chave}: {valor}')
if resultado[0] < resultado[4]['5%']:
    print('A série é estacionária')
else:
    print('A série não é estacionária')

!pip install --upgrade statsmodels
!pip install pmdarima
!pip install --upgrade pmdarima
import numpy as np
from statsmodels.tsa.arima.model import ARIMA
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_squared_error
import itertools
ibov2 = tabelaunificada.drop(columns=["igpm", "selic", "ipca"])
train_size = int(len(ibov2)*0.7)
train_set = ibov2[:train_size]
```

```
test_set = ibov2[train_size:]
print(len(train_set), len(test_set))
p = d = q = range(0, 4)
pdq = list(itertools.product(p, d, q))
aic_values = []
for param in pdq:
    try:
        model = sm.tsa.ARIMA(ibov2["Close"], order=param)
        results = model.fit()
        aic_values.append((param, results.aic))
    except:
        continue

if aic_values:
    best_model = min(aic_values, key=lambda x: x[1])
    print("Best ARIMA model:", best_model[0], " AIC:", best_model[1])
else:
    print("Nenhum modelo ARIMA possível encontrado")

model = ARIMA(train_set['Close'], order=(0,0,3))
result = model.fit()
preds = result.predict(start=test_set.index[0], end=test_set.index[-1])
r2 = r2_score(test_set['Close'], preds)
print('R2:', r2)
mse = mean_squared_error(test_set['Close'], preds)
print('MSE:', mse)

from sklearn.model_selection import train_test_split
X = tabela_normalizada[['ipca', 'igpm', 'selic',]]
y = tabela_normalizada['Close']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3
, random_state=42)

from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Ridge
import matplotlib.pyplot as plt
model = LinearRegression()
ridge_model = Ridge(alpha=1)
ridge_model.fit(X_train, y_train)
y_pred_ridge = ridge_model.predict(X_test)
r2_ridge = ridge_model.score(X_test, y_test)
print("R2 do modelo Ridge:", r2_ridge)
mse_ridge = mean_squared_error(y_test, y_pred_ridge)
print("MSE do modelo Ridge:", mse_ridge)

ridge_model_all = Ridge(alpha=1)
ridge_model_all.fit(X, y)
y_pred_all = ridge_model_all.predict(X)
```

```
ibov3 = pd.DataFrame({'Close': y_pred_all}, index=tabelaunificada.index
)

import numpy as np
import pandas as pd
from keras.models import Sequential
from keras.layers import Dense, LSTM
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, r2_score
model = Sequential()
mod-
el.add(LSTM(units=50, return_sequences=True, input_shape=(X_train.shape
[1], 1)))
model.add(LSTM(units=50))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(X_train, y_train, epochs=30, batch_size=32)
test_predict = model.predict(X_test)
y_pred = model.predict(X_test)
r2 = r2_score(y_test, y_pred)
print(f"R2: {r2:.4f}")
mse = mean_squared_error(y_test, y_pred)
print(f"MSE: {mse:.4f}")


from sklearn.linear_model import Lasso
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3
, random_state=42)
lasso = Lasso(alpha=0.01)
lasso.fit(X_train, y_train)
y_pred = lasso.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print("Erro quadrático médio:", mse)
print("Coeficientes:", lasso.coef_)
r2 = r2_score(y_test, y_pred)
print("R²:", r2)
```