FIAP
Pós Tech em Data Analytics
Tech Challenge 2

Análise e Previsão do Índice IBOVESPA – Relatório Técnico

Utilização de Modelos de Machine Learning

Alunos:

Airton da Silva Cruz Filho
Gustavo Pitarello de Souza
João Paulo Giacherini de Moraes
Victor Moreno Galves Marcondes
Thiago Ribeiro das Costas

Sumário

1. Introdução

O objetivo deste relatório é a análise e a previsão da tendência de fechamento do índice IBOVESPA, identificando se a variação do dia seguinte será de alta ou de baixa. O estudo, que abrange o período de 2020 a 2025, foi desenvolvido por meio de um modelo preditivo.

O documento apresenta a metodologia de aquisição de dados, a análise exploratória, as estratégias de engenharia de atributos, a preparação da base para previsão, a explicação dos modelos utilizados, a comparação entre diferentes algoritmos e a análise dos resultados obtidos. Por fim, são fornecidas as justificativas técnicas para as abordagens adotadas, reforçando a fundamentação metodológica da previsão.

2. Aquisição dos Dados

A base de dados utilizada para a construção do modelo consiste nos registros históricos do índice IBOVESPA, obtidos por meio do site Investing.com. O período de análise abrangeu os dias úteis entre 2020 e 2025, totalizando cinco anos. A escolha desse intervalo teve como objetivo mitigar a influência de eventos atípicos e extremos, como a queda observada no início da pandemia de COVID-19. A quantidade de dados no período é considerada suficiente para evitar riscos de ajuste excessivo (*overfitting*) e ajuste insuficiente (*underfitting*).

3. Análise Exploratória dos Dados

Para compreender o comportamento do IBOVESPA, foi selecionado um período de dez anos (2015 a 2025) para a análise exploratória. Observou-se uma tendência de crescimento do índice ao longo do tempo, mas com variações significativas. Um exemplo é a queda acentuada em 2020 devido à pandemia de COVID-19, seguida por uma leve retração em 2022, influenciada por fatores econômicos globais e internos. O Gráfico 1 ilustra as médias anuais do índice.

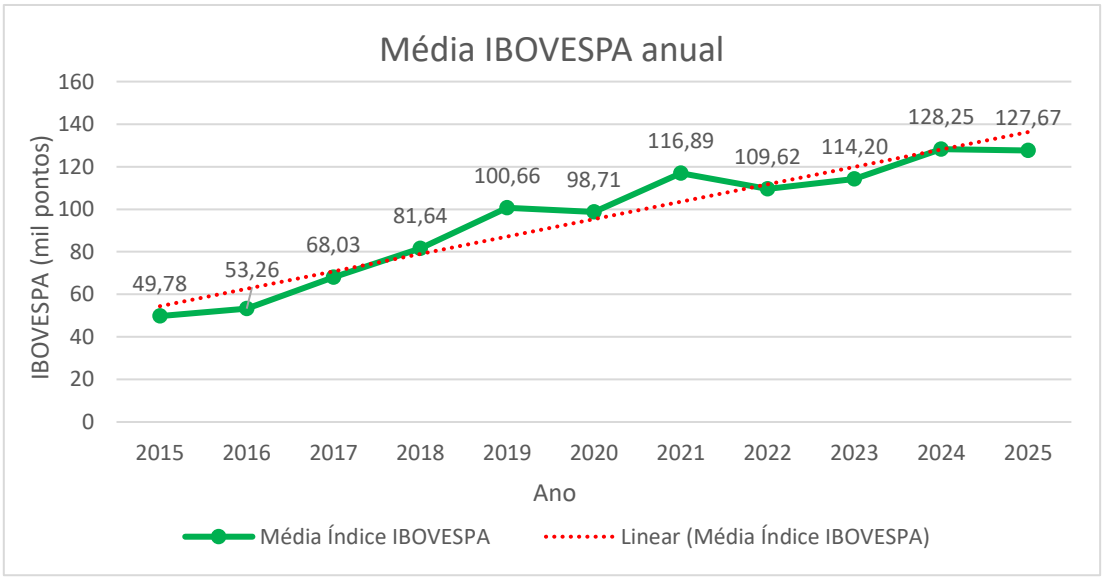

Gráfico 1

A Tabela 1 e o Gráfico 2 detalham as flutuações mensais de 2020 e 2021. Após os primeiros registros da pandemia no Brasil, em fevereiro de 2020, o índice apresentou uma queda significativa, com flutuações de até 41 mil pontos, e sinais de recuperação somente a partir de novembro.

Mês	Média Ibovespa 2020	Média Ibovespa 2021	Diferença (2021 - 2020)
Janeiro	116,90	120,07	3,17
Fevereiro	113,21	117,51	4,30
Março	81,52	113,87	32,35
Abril	77,59	119,25	41,66
Maio	81,45	122,22	40,77
Junho	94,45	128,99	34,54
Julho	101,26	126,03	24,77
Agosto	101,71	120,31	18,60
Setembro	98,58	113,87	15,30
Outubro	97,94	110,09	12,15
Novembro	105,28	104,27	-1,01
Dezembro	115,64	105,90	-9,74
Total	98,79	116,87	18,07

Tabela 1 - Variação mensal do índice IBOVESPA em 2020 e 2021 (mil pontos)

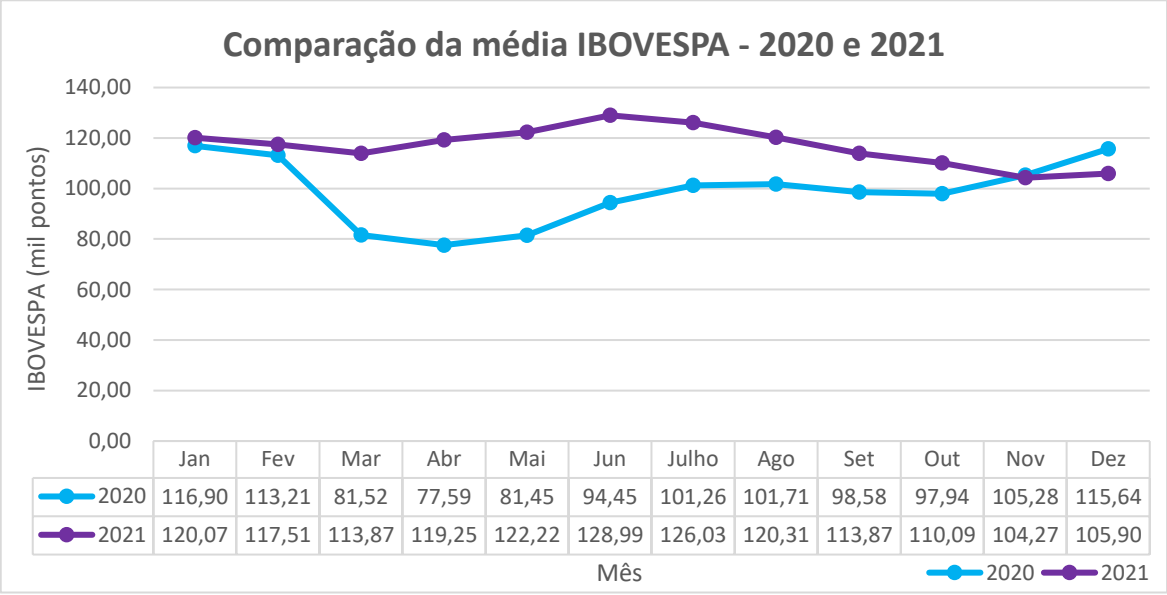

Gráfico 2

Apesar da recuperação pós-pandemia, o ano de 2022 contrariou a tendência de crescimento com uma leve queda, atribuída a fatores como instabilidade política, aumento da taxa de juros (SELIC), tensões geopolíticas e inflação elevada. Essa incerteza resultou na diminuição do valor dos investimentos e na queda do índice ao longo do ano. A partir de 2023, o índice retomou o crescimento, atingindo seu pico em 2024.

4. Preparação dos Dados para Predição: Limpeza, Engenharia de Atributos e Padronização da Escala

A base de dados foi preparada para a modelagem preditiva, seguindo etapas de limpeza, criação de novas variáveis e padronização.

Limpeza dos Dados e Novas Colunas:

- As colunas var% e vol. foram convertidas para o formato numérico, removendo caracteres especiais como vírgulas e letras que representavam grandezas (K, M, B).

```
# Transformando Var% e Vol em float tirando as letras
df['Var%'] = (df['Var%'].astype(str)
              .str.replace(',', '.')
              .str.replace('%', '')
              .astype(float))

df['Vol.'] = (df['Vol.'].astype(str)
              .str.replace(',', '.')
              .str.replace('K', 'e3')
              .str.replace('M', 'e6')
              .str.replace('B', 'e9') # Assuming 'B' means Billion, which is 10^9. Corrected to 'e9'
              .astype(float))
```

Figura 1

- Foram criadas novas colunas com base na data (ano, mes, dia, dia_da_semana e semana_do_ano) e em indicadores técnicos, que funcionam como as "características" que o modelo usará para aprender. Entre os indicadores, estão a “Média Móvel Simples” de 5 e 20 dias (SMA_5, SMA_20), o “Índice de Força Relativa” (RSI) e o “Range” do Dia (Range_Dia).

```
# 1. Converte a coluna 'Data' para o formato de data e define como índice
df2['Data'] = pd.to_datetime(df2['Data'], format='%d.%m.%Y')
df2.set_index('Data', inplace=True)

# --- ETAPA 1: Criar os indicadores técnicos ---

df2['ano'] = df2.index.year
df2['mes'] = df2.index.month
df2['dia'] = df2.index.day
df2['dia_da_semana'] = df2.index.dayofweek
df2['semana_do_ano'] = df2.index.isocalendar().week

# Médias Móveis Simples (SMA) ->  soma os preços de fechamento dos últimos 5 dias e divide por 5.
df2['SMA_5'] = df2['Último'].rolling(window=5).mean()
df2['SMA_20'] = df2['Último'].rolling(window=20).mean()

# Índice de Força Relativa (RSI) -> relação entre as médias de ganhos e as médias de perdas em um determinado período (14 dias)
delta = df2['Último'].diff(1)
ganho = (delta.where(delta > 0, 0)).rolling(window=14).mean()
perda = (-delta.where(delta < 0, 0)).rolling(window=14).mean()
rs = ganho / perda
df2['RSI'] = 100 - (100 / (1 + rs))

# Range do Dia
df2['Range_Dia'] = df2['Máxima'] - df2['Mínima']
```

Figura 2

- Para evitar o vazamento de informação (*data leakage*), foram criadas versões atrasadas das variáveis, utilizando dados do dia anterior (com o sufixo D-1). As colunas originais (Último, Var%, Abertura, Máxima, Mínima e Vol.) foram excluídas do conjunto de variáveis preditoras para garantir que o modelo não utilize dados que não estariam disponíveis no momento da previsão.

```
# --- Criar as features atrasadas (D-1) ---

# Lista de todas as features que vamos usar do dia anterior
features_para_atrasar = [
    'Abertura', 'Máxima', 'Mínima', 'Vol.',
    'SMA_5', 'SMA_20', 'RSI', 'Range_Dia'
]

# Cria uma versão "_D-1" para cada uma
for feat in features_para_atrasar:
    df2[f'{feat}_D-1'] = df2[feat].shift(1)

# Remove a primeira linha que ficará com NaN após o shift
df2.dropna(inplace=True)


# Exibe o DataFrame com as novas colunas
df2.head()
```

Figura 3

Definição do Target, Divisão e Padronização da Escala:

- Como o modelo trabalhará a previsão de forma binária, foi criada uma nova coluna denominada alvo_class, sendo que o valor 1 assume que o dia seguinte o índice IBOVESPA seja Alta, e o valor 0 assume Baixa.

```
# Cria coluna alvo binária

df2.sort_index(inplace=True)
df2['alvo_class'] = (df2['Var%'] > 0).astype(int)
```

Figura 4

- A base foi dividida em um conjunto de treino, com 1.215 registros históricos (29/05/2020 a 14/05/2025), e um conjunto de teste, com os últimos 30 dias de dados (15/04/2025 a 29/05/2025).

```python
# A variável alvo 'y' é a coluna 'alvo_class'
y_train = df_train['alvo_class']
y_test = df_test['alvo_class']

# Lista de colunas para remover (alvo + features originais que vazam dados)
colunas_para_remover = [
    'alvo_class', 'Var%', 'Último',
    'Abertura', 'Máxima', 'Mínima', 'Vol.'
]

# As features 'X' são as colunas restantes
# Usamos as colunas _D-1 que criamos e removemos as originais
X_train = df_train.drop(columns=colunas_para_remover)
X_test = df_test.drop(columns=colunas_para_remover)

# --- Verificação ---
print("--- DADOS DE TREINO ---")
print(f"Shape de X_train: {X_train.shape}")
print(f"Shape de y_train: {y_train.shape}")
print("\n--- DADOS DE TESTE ---")
print(f"Shape de X_test: {X_test.shape}")
print(f"Shape de y_test: {y_test.shape}")

print("\n--- COLUNAS EM X_train ---")
print(X_train.columns.to_list())
```

```
--- DADOS DE TREINO ---
Shape de X_train: (1215, 17)
Shape de y_train: (1215,)

--- DADOS DE TESTE ---
Shape de X_test: (30, 17)
Shape de y_test: (30,)
```

Figura 5

- Para garantir que todas as variáveis tivessem a mesma importância no treinamento, as features dos conjuntos de treino e teste foram padronizadas utilizando o MinMaxScaler, que ajusta os valores para uma mesma faixa (entre 0 e 1).

```python
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Figura 6

5. Modelos Preditivos: Avaliação e Escolha

Inicialmente, para auxiliar na escolha do modelo de previsão, foi realizado o teste de autocorrelação (ACF), com o objetivo de identificar possíveis padrões cíclicos na série e avaliar a dependência temporal da variável-alvo. O teste foi aplicado sobre a variável binária alvo_class (Alta e Baixa), utilizando uma janela temporal de 21 dias úteis.

Conforme ilustrado no Gráfico 3, conclui-se que o comportamento do índice, estar em alta ou baixa, não depende diretamente do valor do dia anterior. Ou seja, o fato de o IBOVESPA ter fechado em alta no Dia 1 não implica necessariamente em uma alta ou baixa no Dia 2, indicando baixa autocorrelação de curto prazo na série binária analisada.

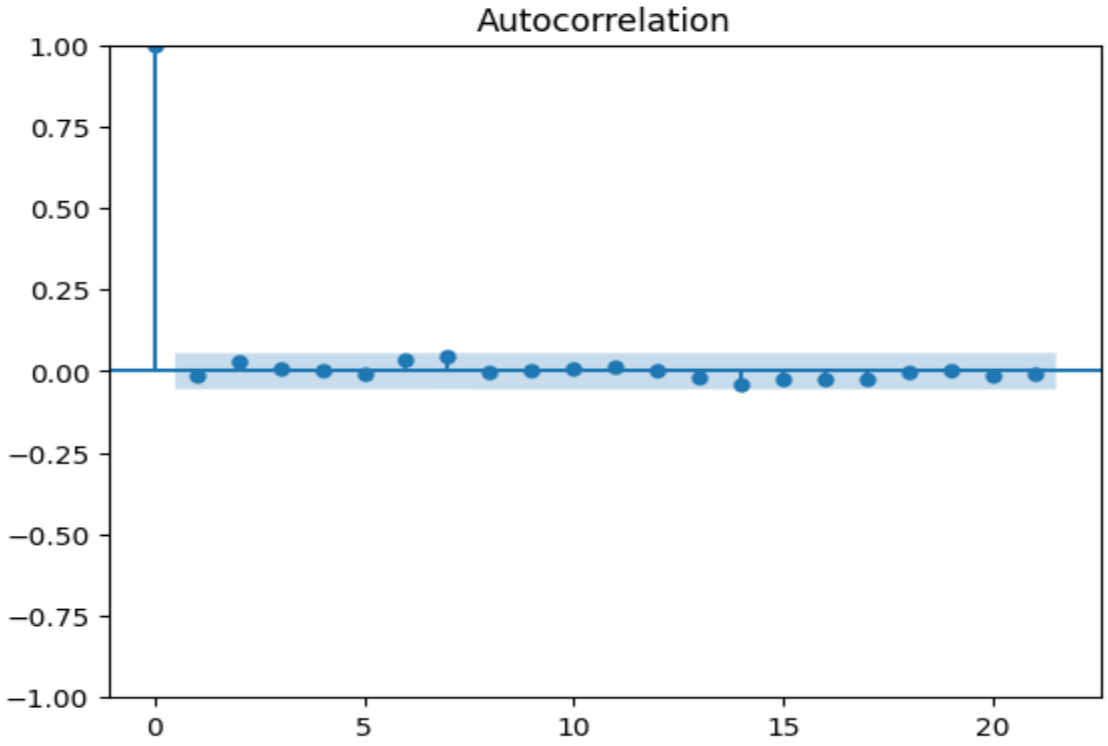

Gráfico 3

Além da autocorrelação, realizamos o teste estatístico de decomposição sazonal, uma técnica de separação da série temporal três componentes principais: tendência, sazonalidade (padrão ao logo do tempo) e resíduos (erros). Conforme os gráficos abaixo, notamos que a "série original", parece um ruído puro, com diversas mudanças abruptas e frequentes, a tendência não exibe um padrão claro, os valores sobem e descem constantemente, a sazonalidade exibe um padrão artificial devido a binaridade da série e o resíduo é alto e aleatório

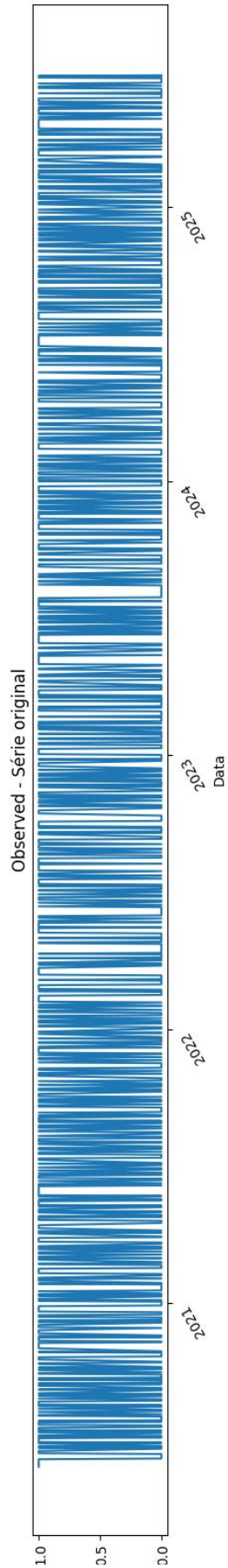

Gráfico 4 - Série Original

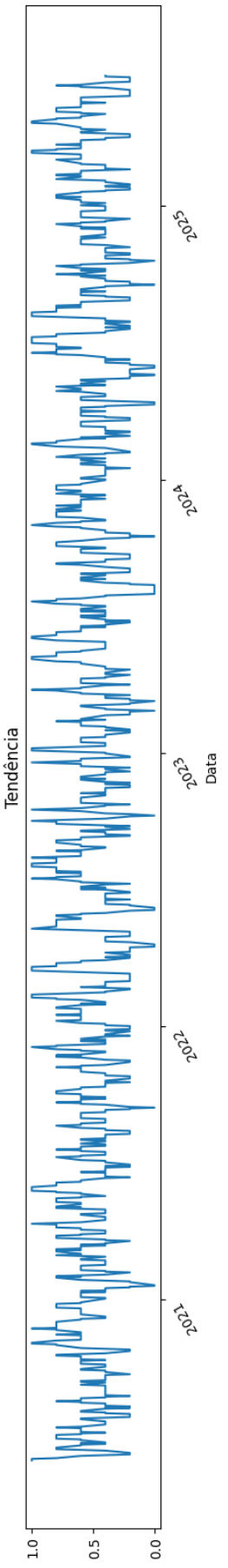

Gráfico 5 - Tendência

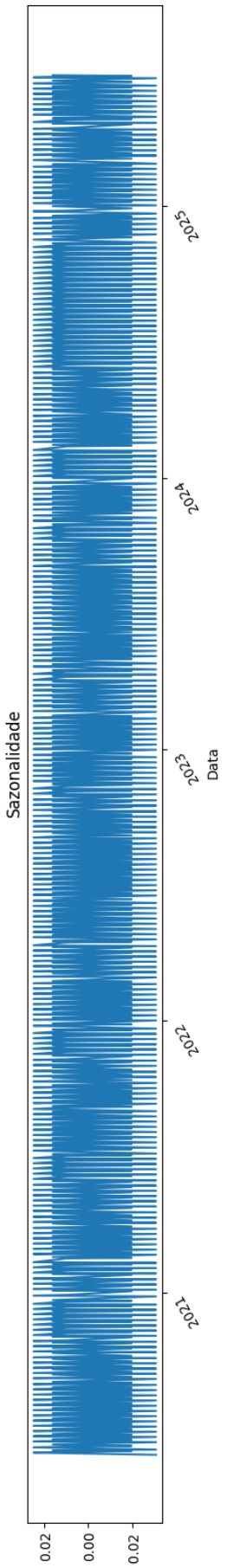

Gráfico 6 - Sazonalidade

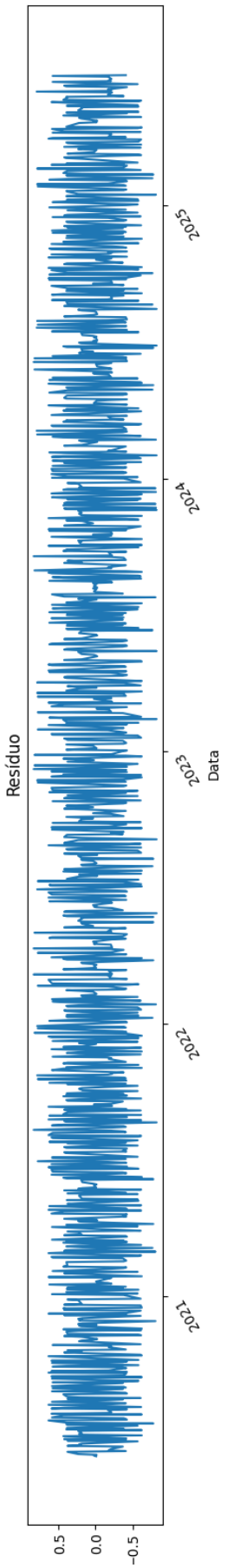

Gráfico 7 - Resíduos

Devido à natureza binária da série, bem como aos resultados obtidos nos testes de autocorrelação e decomposição sazonal, optou-se por descartar modelos que exigem valores contínuos e que dependem fortemente de correlação temporal, tendência e sazonalidade, como é o caso do modelo ARIMA.

Em contrapartida, selecionamos algoritmos que se adaptam melhor às características da base de dados utilizada, especialmente no contexto de classificação binária.

Assim, foram selecionados e testados os seguintes modelos de classificação, mais adequados às características da nossa base de dados:

- Regressão Logística: Um modelo de classificação linear simples, utilizado como base de comparação para avaliar o desempenho de algoritmos mais complexos.
- Árvore de Decisão: Um modelo intuitivo baseado em regras, que permite entender de forma clara o processo de tomada de decisão do algoritmo.
- Random Forest: Um modelo que utiliza várias árvores de decisão para melhorar a precisão da previsão e reduzir o risco de ajuste excessivo (overfitting).
- Gradient Boost e XGBoost: Algoritmos de aprendizado de máquina mais avançados que constroem modelos de forma sequencial, corrigindo os erros das previsões anteriores para otimizar o desempenho.

As métricas utilizadas para a avaliação de desempenho de cada modelo foram: acurácia, precisão, recall e F1-Score.

Para não estender o presente relatório com imagens do código, detalhamos o notebook para melhor entendimento do leitor.

A Tabela 2 e o Gráfico 8 ilustram a comparação entre os resultados obtidos.

Modelo	Acurácia	Precisão	Recall	F1-Score
Regressão Logística	0,60	0,62	0,94	0,75
Árvore de Decisão	0,50	0,64	0,47	0,54
Random Forest	0,70	0,70	0,89	0,79
Gradient Boost	0,56	0,62	0,94	0,75
XGBoost	0,76	0,62	0,94	0,75

Tabela 2

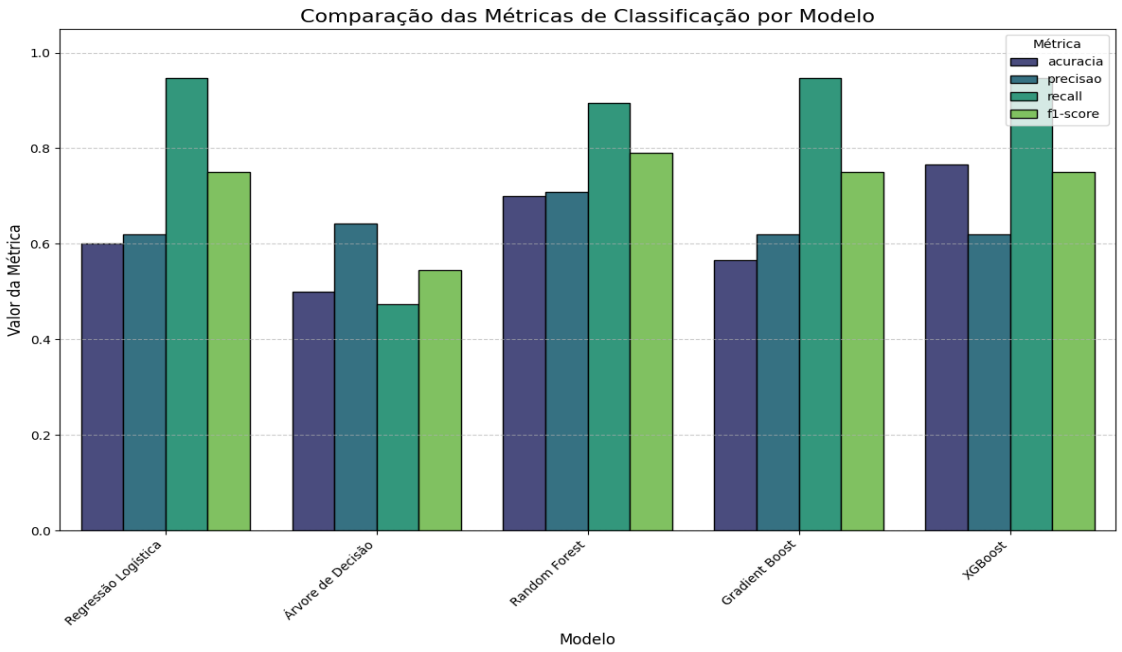

Gráfico 8

A seguir, é apresentada uma breve análise dos resultados obtidos por cada algoritmo.

- **Regressão Logística:** Apresentou uma acurácia de 60%, mas com um baixo desempenho na previsão da classe "queda" (0 acertos em 17 casos reais).

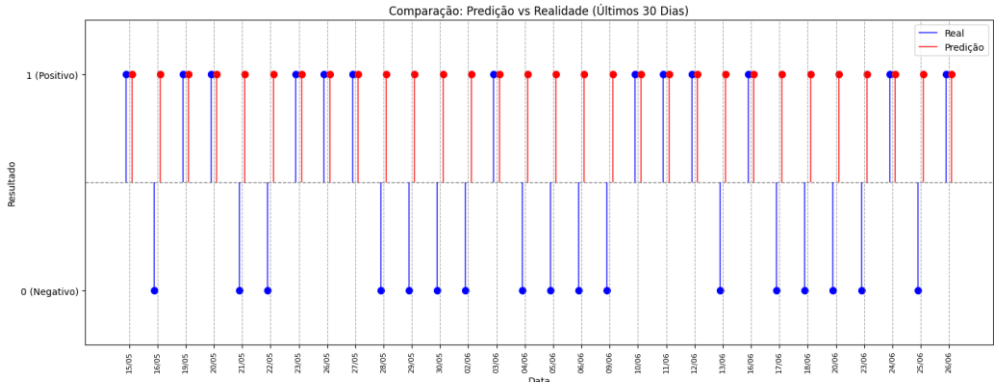

Gráfico 9

- **Árvore de Decisão:** Obteve uma acurácia de 50%, a mais baixa entre os modelos testados, indicando dificuldades em generalizar as previsões.

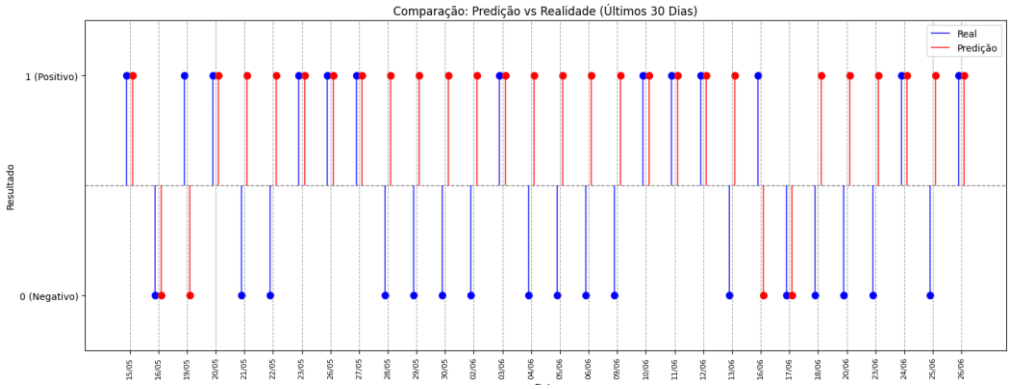

Gráfico 10

- **Random Forest:** Teve um desempenho consideravelmente melhor, com 70% de acurácia. O modelo se mostrou robusto, principalmente na identificação da classe "alta", com um recall de 89%.

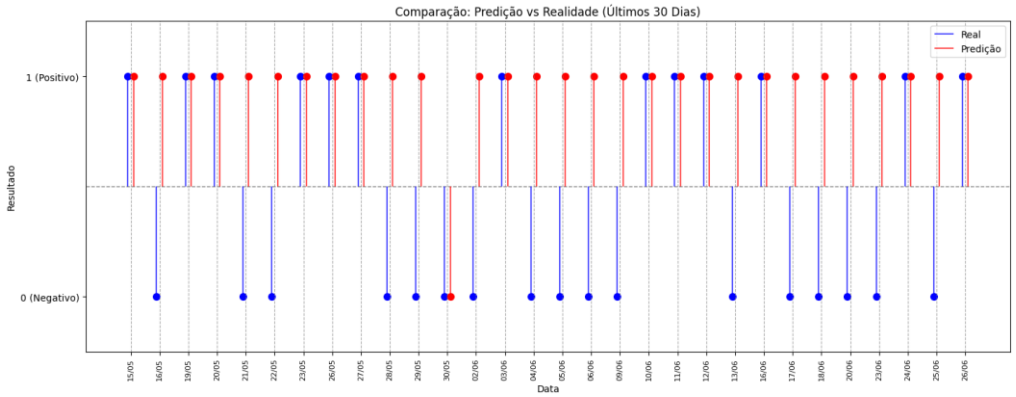

Gráfico 11

- **Gradient Boost:** Com 56% de acurácia, teve um desempenho inferior ao do Random Forest.

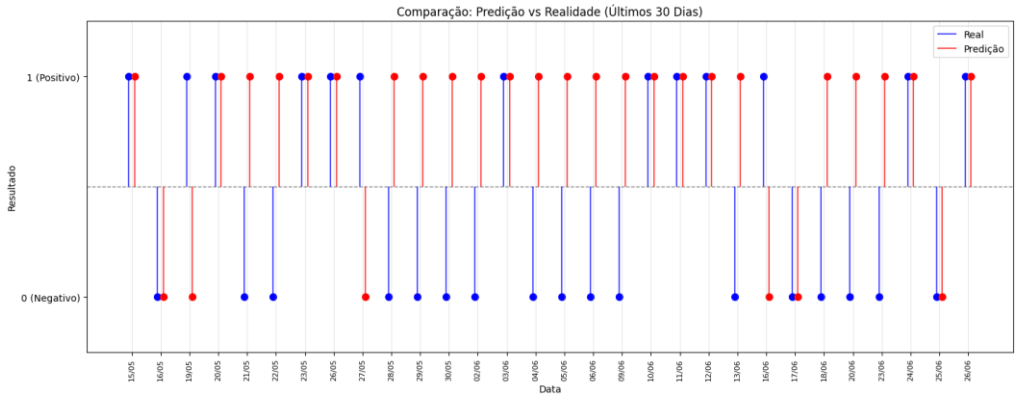

Gráfico 12

- **XGBoost:** Foi o modelo de destaque, alcançando a maior acurácia (76%). Por conta do melhor desempenho geral nas métricas, o XGBoost foi o algoritmo escolhido para a análise detalhada.

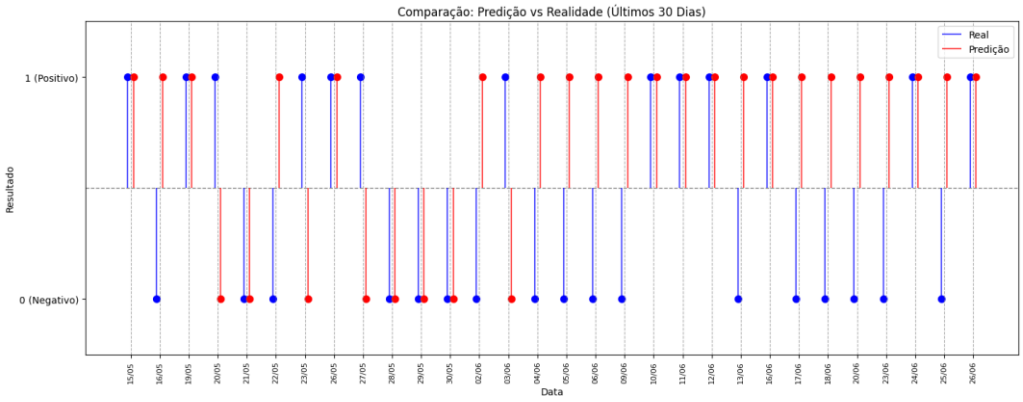

Gráfico 13

6. Análise Detalhada do Modelo Escolhido (XGBoost)

Para uma avaliação aprofundada do modelo XGBoost, foi analisada a importância de cada variável, feita uma comparação gráfica entre valores previstos e reais, e gerada uma matriz de confusão e um relatório de classificação.

Importância das Features:

O Gráfico 14 mostra a contribuição de cada variável para o desempenho do modelo. As variáveis Vol., D-1, Abertura_D-1, Máxima_D-1 e Mínima_D-1 (dados do dia anterior) foram as mais importantes para a previsão.

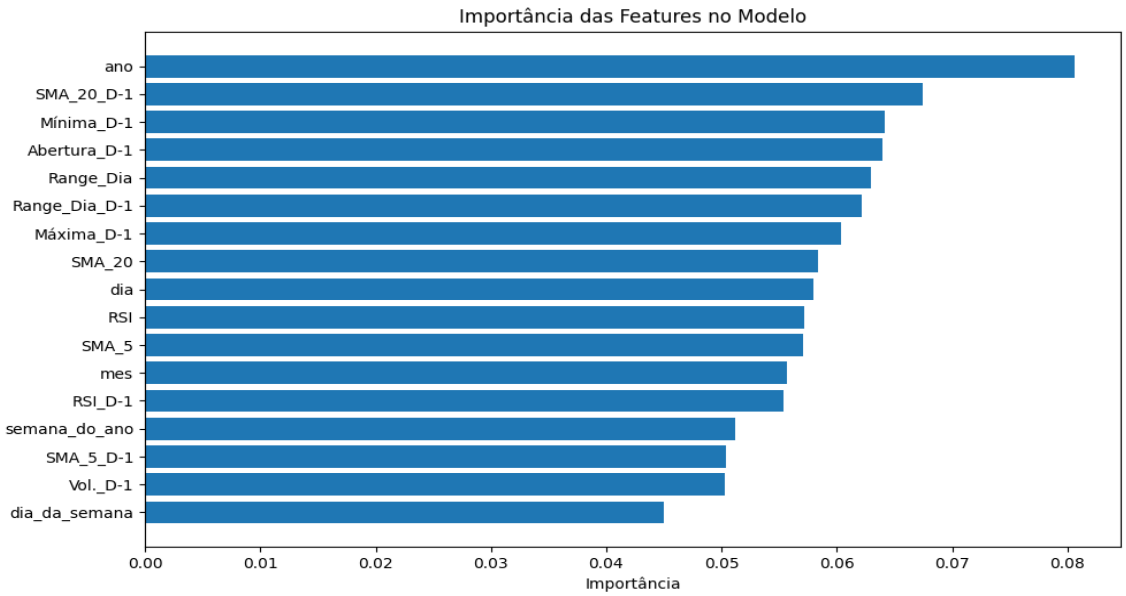

Gráfico 14

Matriz de Confusão:

A Matriz de Confusão (Gráfico 15) apresenta o resumo das previsões. A diagonal principal indica os acertos do modelo:

- Previsões corretas de queda: O modelo identificou corretamente 4 das 11 quedas que ocorreram.
- Previsões corretas de alta: O modelo acertou 19 das 19 altas, ou seja, 100% dos casos.

A outra diagonal mostra os erros de previsão:

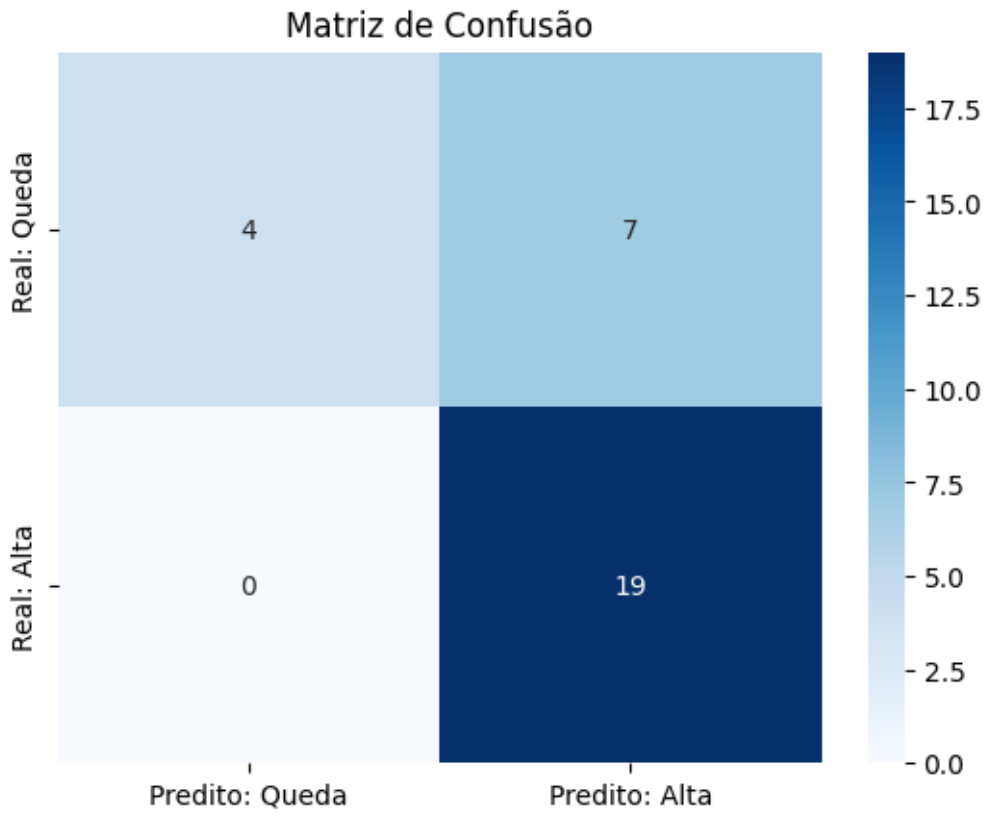

Gráfico 15

- Falsos positivos: O modelo previu uma queda em 7 ocasiões, mas o IBOVESPA fechou em alta.

Relatório de Classificação:

A Tabela 3 apresenta um relatório completo das métricas para cada classe.

- Classe "Queda" (0): A precisão foi de 100%, mas o recall foi de apenas 36%, indicando que o modelo é confiável quando prevê uma queda, mas tem dificuldade em detectar todos os casos de queda.
- Classe "Alta" (1): A precisão foi de 73%, e o recall foi de 100%.
- Métricas Agregadas: A acurácia geral do modelo foi de 76%.

Métrica	Queda	Alta	Média	Média ponderada	Acurácia
Precisão	1	0,73	0,87	0,83	-
Recall	0,36	1	0,68	0,77	-
F1-Score	0,53	0,84	0,69	0,73	0,76
Support	11	19	30	30	30

Tabela 3

7. Conclusão

Com base na análise e nos resultados dos modelos testados, o algoritmo XGBoost se destacou como a melhor opção para a previsão da tendência do IBOVESPA, especialmente por sua alta acurácia de 76%.

O modelo se mostrou mais eficiente em prever a tendência de alta (com 100% de recall), enquanto a previsão da tendência de queda apresentou um recall mais baixo, indicando espaço para futuras melhorias. A importância das variáveis do dia anterior (como volume, abertura, máxima e mínima) reforça a dependência das previsões em relação aos dados recentes do mercado.

8. Links para Vídeo e Repositório GitHub

Vídeo: https://www.youtube.com/watch?v=vOflDnOZcMQ

Repositório GitHub: https://github.com/trcosta97/fiap-tech-challenge2/blob/main/README.md