**Forecasting**

Neste estudo, o objetivo da previsão de vendas foi otimizar o planejamento de estoque, logística e estratégias de reposição de produtos, proporcionando uma visão prospectiva para os próximos 28 dias (4 semanas). As previsões foram realizadas especificamente para diversas lojas e produtos, buscando reduzir erros de estoque e melhorar a alocação de recursos.

**Pré-processamento dos Dados**

O pré-processamento dos dados envolveu múltiplas etapas para garantir a qualidade e relevância das informações utilizadas nos modelos. Inicialmente, valores ausentes nas vendas foram preenchidos com zero para manter a consistência dos dados. Em seguida, indicadores de eventos sazonais e temporais foram adicionados, incluindo marcadores para o Natal (is\_christmas), feriados (is\_holiday), dia da semana (day\_of\_week) e finais de semana (is\_weekend), refletindo padrões de demanda específicos. Outliers foram detectados e removidos usando o método do z-score, assegurando que valores atípicos não distorcessem os resultados. Além disso, itens com alta proporção de vendas zero foram filtrados para melhorar a eficiência dos modelos. Essas variáveis e ajustes permitiram capturar tendências sazonais e comportamentos específicos das vendas ao longo do tempo.

**Modelos de Previsão**

Optou-se por uma abordagem híbrida, utilizando três modelos principais: Prophet, SARIMA e XGBoost. Cada modelo foi escolhido por suas características específicas de adequação à natureza dos dados e aos requisitos da tarefa:

* **Prophet**: Escolhido pela robustez na captura de sazonalidades e tendências, especialmente em séries temporais com eventos sazonais como feriados, que impactam diretamente o comportamento das vendas. A capacidade de incorporar feriados como variáveis exógenas foi um diferencial importante para períodos de alta variabilidade.
* **SARIMA**: Aplicado devido à sua capacidade de modelar sazonalidades, principalmente em dados com padrões semanais e mensais. O SARIMA se ajusta bem aos dados onde dependências sazonais são predominantes, como em diversas categorias de produtos de supermercado.
* **XGBoost**: Integrado para capturar padrões não lineares complexos e interações entre variáveis, especialmente relevante em cenários de grandes variações de vendas, onde fatores externos e categorias de produtos desempenham papéis significativos.

**Avaliação dos Modelos**

A precisão de cada modelo foi avaliada com base em métricas comuns em previsões de séries temporais. A tabela a seguir resume as métricas médias de desempenho para cada modelo:

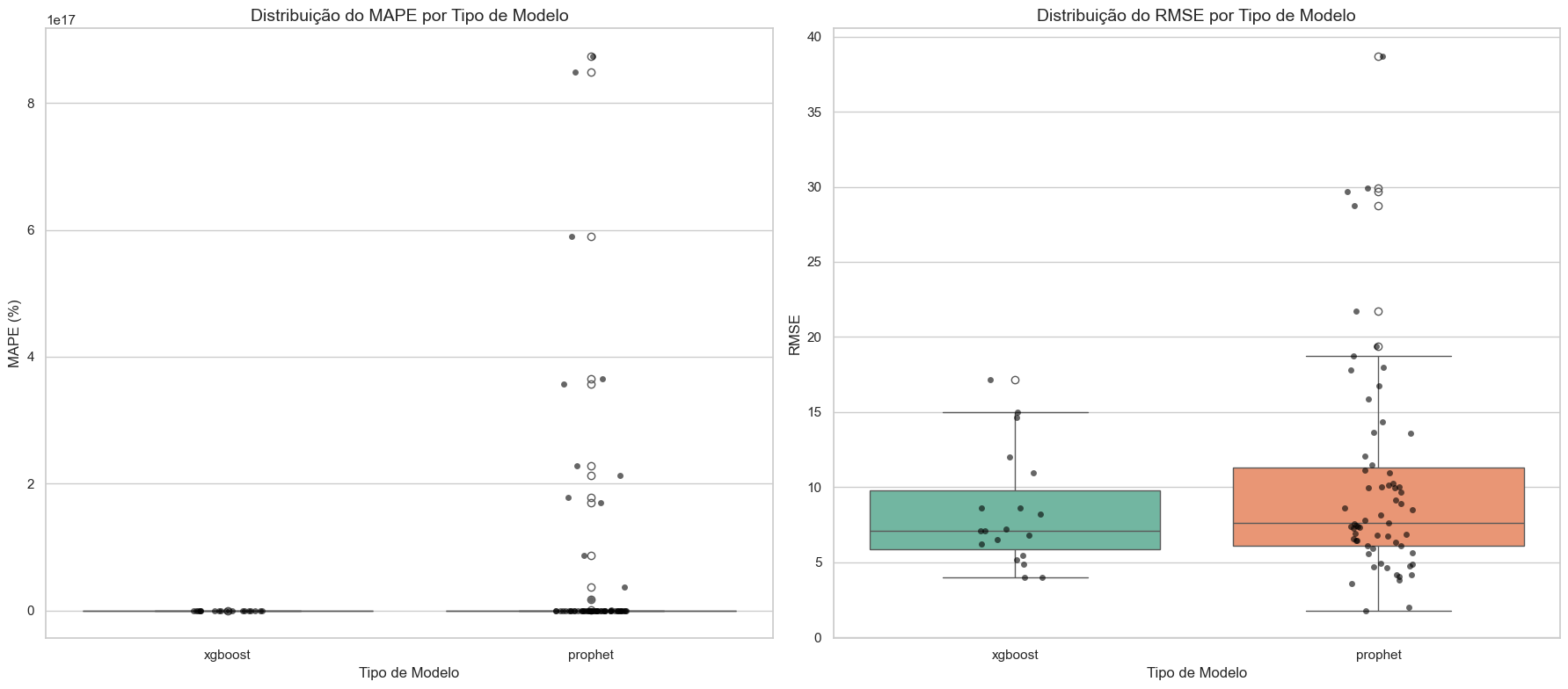
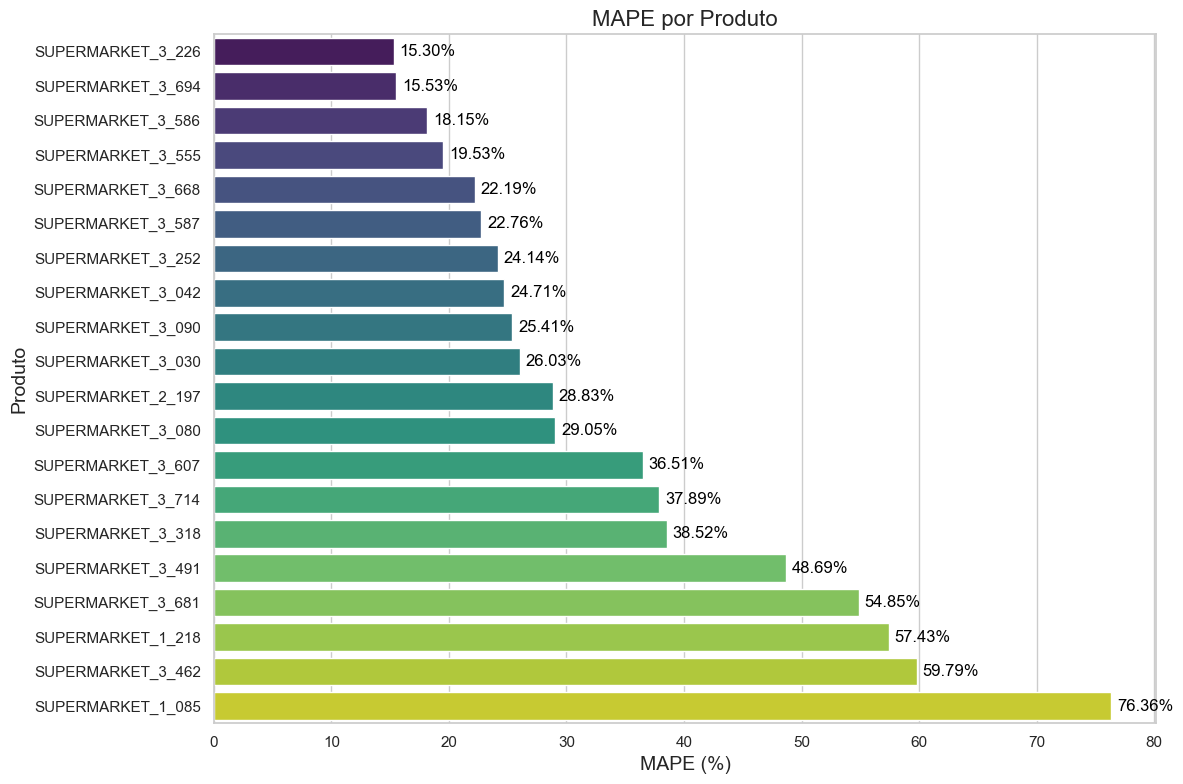
| **Modelo** | **RMSE Médio** | **MSE Médio** | **MAE Médio** | **MAPE Médio** | **R² Médio** |
| --- | --- | --- | --- | --- | --- |
| Prophet | 12.84 | 256.42 | 9.87 | 1.386e+17 | -0.20 |
| SARIMA | 9.92 | 131.69 | 8.83 | 20.79 | -0.16 |
| XGBoost | 8.39 | 87.50 | 6.40 | 23.43 | 0.26 |

**Notas sobre as Métricas:**

* **RMSE (Root Mean Squared Error)**: Indica a dispersão dos erros entre as previsões e os valores reais. O XGBoost apresentou o menor RMSE médio (8.39), sugerindo maior precisão geral.
* **MAE (Mean Absolute Error)**: Mede o erro médio absoluto entre as previsões e os valores reais. Novamente, o XGBoost teve o melhor desempenho com um MAE médio de 6.40.
* **MAPE (Mean Absolute Percentage Error)**: Expressa o erro absoluto médio em termos percentuais. O Prophet apresentou um MAPE médio elevado devido a valores extremos em alguns produtos com vendas muito baixas, o que distorce a média.
* **R² (Coeficiente de Determinação)**: Indica o quão bem os dados previstos se ajustam aos dados reais. Valores próximos de 1 representam um bom ajuste. O XGBoost apresentou um R² médio positivo (0.26), enquanto os outros modelos tiveram valores próximos ou abaixo de zero, indicando menor capacidade explicativa.

Os resultados indicaram que o XGBoost foi especialmente eficaz em capturar padrões complexos e interações entre variáveis, resultando em menor erro médio. O Prophet foi eficiente em capturar sazonalidades relacionadas a feriados, mas apresentou maior erro em produtos com vendas irregulares. O SARIMA teve desempenho sólido em produtos com padrões sazonais regulares.

**Gráficos Comparativos**

* **Distribuição do MAPE e RMSE por Modelo**: Os gráficos a seguir mostram a variação das métricas de MAPE e RMSE para cada modelo, oferecendo uma visão das discrepâncias de erro entre as previsões.
* 
* **MAPE por Produto**: Um gráfico de barras exibe o MAPE para diferentes produtos, destacando aqueles com menor erro percentual, o que indica alta precisão nas previsões.
* 

**Análise Comparativa entre Lojas e Produtos**

As previsões de vendas foram realizadas para várias lojas, com destaque para as cinco com maior volume de vendas:

* **Tribeca** liderou com 7.926 vendas previstas, seguida por **Roxbury** (6.717), **Back Bay** (5.942), **Queen Village** (5.753) e **South End** (4.893). Essas lojas demonstraram uma demanda consistente, especialmente em produtos específicos que contribuem significativamente para o total de vendas.

Entre os produtos, os itens de maior venda foram:

| **Produto** | **Vendas Previstas** |
| --- | --- |
| SUPERMARKET\_3\_090 | 12.603 |
| SUPERMARKET\_3\_586 | 10.564 |
| SUPERMARKET\_3\_252 | 6.425 |
| SUPERMARKET\_3\_555 | 3.825 |
| SUPERMARKET\_3\_226 | 3.489 |

Esses produtos representam uma grande parte das vendas totais e são fundamentais para a definição de estratégias de reposição e estoque.

**Conclusão**

A aplicação de múltiplos modelos de previsão na análise de vendas da DSMarket proporcionou uma perspectiva abrangente e confiável para o planejamento estratégico. O XGBoost demonstrou desempenho superior em termos de precisão geral, capturando efetivamente padrões não lineares complexos. O Prophet foi valioso para incorporar efeitos sazonais e eventos especiais, enquanto o SARIMA complementou a análise em produtos com sazonalidade regular.

Recomenda-se a inclusão de dados econômicos e informações sobre campanhas de marketing para tornar as previsões ainda mais robustas. Além disso, a atualização contínua dos modelos com novos dados e o ajuste de parâmetros conforme a demanda são essenciais. Dessa forma, as previsões de vendas tornam-se um recurso crucial para tomadas de decisões estratégicas, melhorando a eficiência das operações e minimizando incertezas no gerenciamento de estoque da DSMarket.

**Anexo: Detalhes Técnicos do Pré-processamento e Modelagem**

*Este anexo inclui detalhes adicionais sobre o código e métodos utilizados no processo de pré-processamento e modelagem.*

**Pré-processamento dos Dados**

O código implementa um pipeline de pré-processamento que inclui as seguintes etapas:

1. **Tratamento de Valores Ausentes**: Os valores ausentes na coluna de vendas foram preenchidos com zero para evitar problemas durante a modelagem.

**Adição de Variáveis de Evento e Temporais**: Funções como add\_event\_features adicionaram colunas para indicar se uma data corresponde ao Natal (is\_christmas), a outros feriados (is\_holiday), o dia da semana (day\_of\_week) e se é fim de semana (is\_weekend).  
python  
Copiar código  
def add\_event\_features(df):

df['is\_christmas'] = df.apply(lambda x: 1 if x['date'].month == 12 and x['date'].day in [23, 24, 26] else 0, axis=1)

df['is\_holiday'] = df.apply(lambda x: 1 if x['date'].month == 12 and x['date'].day == 25 else 0, axis=1)

df['day\_of\_week'] = df['date'].dt.dayofweek

df['is\_weekend'] = df['day\_of\_week'].isin([5, 6]).astype(int)

return df

**Detecção e Remoção de Outliers**: Utilizando o método do z-score, foram identificados e removidos valores atípicos que poderiam distorcer os resultados dos modelos.  
python  
Copiar código  
def detect\_and\_remove\_outliers(data: pd.DataFrame, column='sales', z\_thresh=3):

data = data.copy()

data['z\_score'] = (data[column] - data[column].mean()) / data[column].std()

cleaned\_data = data[data['z\_score'].abs() <= z\_thresh]

cleaned\_data = cleaned\_data.drop(columns=['z\_score'])

return cleaned\_data

**Filtragem de Itens com Alta Proporção de Vendas Zero**: Itens com mais de 90% de vendas zero foram identificados e excluídos para melhorar a eficiência dos modelos.  
python  
Copiar código  
def filter\_zero\_sales(data: pd.DataFrame, threshold=0.9) -> bool:

zero\_sales\_ratio = (data['sales'] == 0).mean()

if zero\_sales\_ratio > threshold:

logging.warning(f"Item com {zero\_sales\_ratio\*100:.2f}% de vendas zero. Considerar remoção ou tratamento especial.")

return False

return True

**Agregação dos Dados**: Os dados foram configurados para uma frequência diária, preenchendo datas ausentes e garantindo uma série temporal contínua.  
python  
Copiar código  
sales\_data = sales\_data.asfreq('D').fillna(0)

**Modelagem e Avaliação**

O pipeline de modelagem incluiu a aplicação de três modelos diferentes para cada item e loja:

**Prophet**: Após otimização de hiperparâmetros, o modelo foi ajustado e suas previsões foram avaliadas. O modelo foi salvo para uso futuro.  
python  
Copiar código  
forecast\_prophet, model\_type, prophet\_params, prophet\_model = fit\_and\_forecast\_prophet\_optimized(

train\_df, test\_df, steps, regressors, n\_trials=50

)

**SARIMA**: O modelo SARIMA foi otimizado e ajustado aos dados, com previsões avaliadas e o modelo salvo.  
python  
Copiar código  
forecast\_sarima, model\_type, sarima\_params, sarima\_model = fit\_and\_forecast\_sarima\_optimized(

train\_data, test\_data, steps

)

**XGBoost**: Foi ajustado um modelo XGBoost para capturar padrões não lineares, com as previsões sendo avaliadas e o modelo salvo.  
python  
Copiar código  
forecast\_xgboost, model\_type, xgboost\_params, xgboost\_model = fit\_and\_forecast\_xgboost(

train\_df, test\_df, steps, regressors

)

Para cada modelo, as previsões foram comparadas com os dados de teste, e métricas como RMSE, MSE, MAE, MAPE, R², SMAPE e MASE foram calculadas. O melhor modelo para cada item e loja foi selecionado com base no menor MAPE.

**Armazenamento e Visualização**

Os modelos treinados foram salvos para reutilização, e os resultados das previsões foram armazenados. Gráficos de previsão foram gerados para visualização do desempenho dos modelos.

python

Copiar código

# Salvar o gráfico de previsão do melhor modelo

forecast\_df = pd.DataFrame({

'date': test\_df['ds'],

'forecast\_sales': best\_forecast,

'actual\_sales': test\_df['y'].values,

'store': store,

'item': item

})

plot\_path = os.path.join(BASE\_DIR, 'metrics', f"forecast\_{store}\_{item}\_{best\_model\_type}.png")

plot\_forecast(forecast\_df, save\_path=plot\_path)

logging.info(f"Gráfico salvo em {plot\_path}")