UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO

Murilo Lima Ribeiro

Indústria de Fertilizantes: aplicação de técnicas de aprendizado dinâmico na previsão da demanda de matérias-primas

São Carlos

2021

Murilo Lima Ribeiro

Indústria de Fertilizantes: aplicação de técnicas de aprendizado dinâmico na previsão da demanda de matérias-primas

Trabalho de conclusão de curso apresentado ao Centro de Ciências Matemáticas Aplicadas à Indústria do Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, como parte dos requisitos para conclusão do MBA em Ciências de Dados.

Área de concentração: Ciências de Dados

Orientador: Prof. Dr. Antonio Castelo Filho

São Carlos

2021

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi, ICMC/USP, com os dados fornecidos pelo(a) autor(a)

Ribeiro, Murilo Lima
S856m Indústria de Fertilizantes: aplicação de técnicas de aprendizado dinâmico na previsão da demanda de matérias-primas / Murilo Lima Ribeiro ; orientador Antonio Castelo Filho. – São Carlos, 2021.
105 p. : il. (algumas color.) ; 30 cm.

Monografia (MBA em Ciências de Dados) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2021.

1. Ciência de Dados. 2. Aprendizado Dinâmico. 3. Séries temporais I. Filho, Antonio Castelo, orient. II. Título.

Murilo Lima Ribeiro

Indústria de Fertilizantes: aplicação de técnicas de aprendizado dinâmico na previsão da demanda de matérias-primas

Trabalho de conclusão de curso apresentado ao Centro de Ciências Matemáticas Aplicadas à Indústria do Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, como parte dos requisitos para conclusão do MBA em Ciências de Dados.

Data de defesa: 05/03/2022

Comissão Julgadora:

Prof. Dr. Antonio Castelo Filho
Orientador

Prof. Fernando Santos
Convidado

São Carlos
2021

"E tu, Madama Lucrécia, flor dos Bórgias, se um poeta te pintou como a Messalina católica, apareceu um Gregorivius incrédulo que te apagou muito essa qualidade, e se não vieste a lírio, não ficaste pântano. Eu deixo-me estar entre o poeta e o sábio"

Machado de Assis

RESUMO

RIBEIRO, M. L. **Indústria de Fertilizantes: aplicação de técnicas de aprendizado dinâmico na previsão da demanda de matérias-primas**. 2021. 105p. Monografia (MBA em Ciências de Dados) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2021.

A tomada de decisão na indústria de fertilizantes é um processo multidepartamental, que agrega diversas área em torno do ciclo do Sales and Operations Planning (ou S&OP). Torna-se cada vez mais comum a utilização de técnicas mais robustas com objetivo de antecipar valores históricos. Em especial, para o caso desse trabalho, há foco no forecast de volume de entregas de fertilizante e consumo das matérias-primas cloreto de potássio e ureia. Define-se formalmente as séries temporais como a realização de processos estocásticos de uma variável aleatória ao longo do tempo. Utilizam-se técnicas de visualização de dados e estimação de modelos de previsão, para os quais se afere sua capacidade preditiva. De modo geral, para a fábrica de São Luís da Eurochem Fertilizantes Tocantins, é possível estimar o volume de entregas diário de fertilizantes através de um modelo **sARIMA(6,1,0)(1,0,2)[7]** com MAPE $= 22,99\%$. Adicionalmente, as entregas mensais podem ser descritas utilizando-se **sARIMA(1,0,0)(0,1,1)[12]** com MAPE $= 13,60\%$. Para o consumo de cloreto de potássio, o modelo **sARIMA(0,1,0)(1,1,0)[12]** apresentou os melhores resultados preditivos com MAPE $= 12,78\%$. Por fim, para consumo de ureia, modelos sARIMA performaram mal, porém, uma suavização por Holt-Winter teve boa aderência e, quando usada para forecast, apresentou MAPE $= 29,45\%$.

Palavras-chave: fertilizantes; safra; safrinha; séries temporais; aprendizado dinâmico; sARIMA; S&OP; forecast; predição; Holt-Winters

ABSTRACT

RIBEIRO, M. L. **Fertilizer Industry: application of dynamic learning techniques for raw material demand prediction**. 2021. 105p. Monografia (MBA em Ciências de Dados) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2021.

The decision making in the fertilizer industry is a multidepartment process which aggregates several areas around the cycle of Sales and Operations Planing (S&OP). It is becoming more common by the day the use of more robust forecasting techniques to anticipate historical values. For the purposes of this work, there is focus in the forecast of fertilizer deliveries volumes as well as the consumption of raw materials such as potassium chloride and urea. Formally, a time series is defined as the a stochastic process in which a random variable occurs through time. Data visualization is used alongside predictive models estimation to determine predictive capability. In general, for Eurochem's São Luís factory, it is possible to estimate daily deliveries through a **sARIMA(6,1,0)(1,0,2)[7]** model with MAPE $= 22,99\%$. Additionally, monthly deliveries can be described using **sARIMA(1,0,0)(0,1,1)[12]** model with MAPE $= 13,60\%$. With respect to potassium chloride consumption, a **sARIMA(0,1,0)(1,1,0)[12]** proved itself as a good fit for predictive purposes, yielding MAPE $= 12,78\%$. Lastly, urea consumption was not well modelled by sARIMA models, however, smoothing via Holt-Winters adhered to actual data and when used to forecast, performed with MAPE $= 29,45\%$.

Keywords: fertilizer; crop; winter crop; time series; dynamic learning; sARIMA; S&OP; forecast; prediction; Holt-Winters

SUMÁRIO

LISTA DE FIGURAS

1 INTRODUÇÃO

1.1 Motivação

Em 2019, de acordo com o indicador *Agriculture, Forestry and Fishing, Value Added (% of GDP)*[1], o Brasil apresentou uma participação de 4,4% do setor agrícola no PIB nacional, crescimento de aproximadamente 11% em relação ao valor de 2015 (The World Bank, 2021). A despeito da crescente participação do setor na economia, observa-se um cenário de insuficiência no abastecimento interno de fertilizantes minerais, gerando dependência das importações para subsidiar o crescimento na demanda dos produtores agrícolas (OGINO et al., 2021). Ademais, estima-se um aumento médio no preço dos fertilizantes, entre 0,8% a 3,6% por ano, de 2005 a 2050 (BRUNELLE et al., 2015).

Considerada a realidade que permeia o abastecimento do setor de produção de insumos agrícolas, em especial o de produção de fertilizantes, a crescente nos preços de matérias-primas e a dependência dessa indústria em torno da disponibilidade desses insumos criam, portanto, um ponto focal estratégico: previsibilidade da demanda das matérias-primas essenciais. Fatores intrínsecos ao negócio, tais como sazonalidade e dependência de fornecedores externos, são elementos comuns de ruptura em modelos tradicionais de séries temporais (ANDRADE, 2020). Portanto, justifica-se a propositura de uma nova abordagem metodológica que objetive a redução do erro médio percentual absoluto na previsão de consumo de matérias-primas.

A necessidade de um posicionamento competitivo para as grandes indústrias de fabricação de fertilizantes é um desafio conjunto entre perpectivas de manufatura, suprimentos e logística. Nesse sentido, a Eurochem Fertilizantes Tocantins movimentou-se ativamente consolidando um *Enterprise Resource Planning (ERP)* robusto que integra todas suas unidades operacionais e unificando as estratégias de diferentes departamentos responsáveis pela tomada de decisão, simplificando as etapas de planejamento. Pela definição de Thomé et al. (2012), *Sales and Operations Planning (S&OP)* é a unificação entre os diferentes planos de negócio, balanceando suprimento e demanda e construindo pontes entre departamentos. A existência de ciclos de S&OP, continuam Thomé et al. (2012), está positivamente correlacionado com a performance da companhia. A robustez da metodologia de previsão de demanda é um facilitador do processo de implementação

[1] O indicador Agriculture, Forestry and Fishing, Value Added (% of GDP) é uma estatística compilada pelo Banco Mundial que considera integralmente o agronegócio, agrupando "silvicultura, caça, pesca, cultivo de safras e criação de gado. É composta pelo valor acumulado da produção de um setor após descontados todos seus produtos intermediários. Não leva em consideração deduções por depreciação ou degradação de recursos naturais. Utiliza-se como metódo de agregação a média ponderada das variáveis que contribuem para o indicador (The World Bank Data Catalog, 2021)

(PEDROSO; SILVA; TATE, 2016). Fiorucci et al. (2016) exaltam a importância de métodos de estimativas acurados como promotor relevante de um S&OP eficiente.

1.2 Objetivos

Wallace (2012) explica que durante a fase de Planejamento de Demanda dentro do ciclo do S&OP, as previsões devem ser executadas e interpretadas tão logo não representem o melhor prognóstico do futuro, restando à gerência de vendas e marketing a incumbêndia de utilizar seus conhecimentos específicos para validar sua acurácia. O objetivo específico deste trabalho é suplementar a visão de demanda através de previsões de análise temporal e independentes da área de vendas. Para isso, a demanda será dividida entre unidades fabris e tipo de matéria-prima, escolhendo os ingredientes que compõe majoritariamente a demanda de produção. Os resultados de previsão, portanto, serão utilizados para modelar a etapa de Demanda Irrestrita das reuniões mensais e serão utilizados na granularidade semanal para antever o nível de entregas executado, avaliando se o plano do ciclo anterior terá condições de ser executado. A unidade fabril de São Luís será utilizada como referência para os estudos de previsão.

1.3 Organização do Trabalho

Este trabalho está divido em seções que cumprem, em linhas gerais, os seguintes objetivos: elucidar a natureza das séries temporais, aplicar técnicas computacionais para descrever os fenômenos de demanda da Eurochem Fertilizantes Tocantins na granulidade de unidade fabril e matéria-prima, gerar modelos que representem de maneira satisfatória essa demanda, aferir o erro de previsão e implementar ao ciclo do S&OP. Cada um desses objetivos está melhor explicado adiante.

1.3.1 Natureza das séries temporais

O capítulo 2 se dedica a fazer uma revisão bibliográfica cujo intuito é compreender parte do conhecimento corrente de tais fenômenos. Busca-se descrevê-los e entendê-los em sua forma mais simples. Portanto, inicialmente, na seção 2.1, uma visão mais ampla da literatura anteriores introduz o problema e inicia o tom do trabalho, além de criar uma trajetória comum entre teoria e expectativas. Já na seção 2.2, as definições tradicionais da representação de séries temporais são apresentadas formalmente, definindo a linguagem matemática necessária. Além disso, inicia-se também a discussão das métricas de erro que classificam diferentes modelos. A seção 2.3 se dedica a descrever modelos sazonais auto-regressivos integrados de médias móveis (sARIMA). A seção 2.4, por outro lado, aborda um modelo utilizado como referência na validação de performance de muitas séries temporais - o método Theta. A seção 2.5 trata de alguns outros métodos alternativos, como

o Holt-Winters, Rede Neurais Recorrentes (RNN) e, por fim, literatura recente dissertando acerca de modelos híbridos ARIMA-RNN.

1.3.2 Aplicação na descrição dos fenômenos de demanda

A metodologia utilizada por este trabalho está disposta no capítulo 3. A seção 3.1 disserta acerca do pré-trabalho necessário para a extração das informações dos sistemas corporativos. Explica-se resumidamente a estrutura de dados da empresa, as técnicas de SQL para geração de relatórios e o tratamento de base de dados necessário para produzir as informações que serão utilizadas. São dois grandes grupos de dados: produção por matéria-prima e entregas de faturamento. Demais tópicos da seção 3 delongam-se na descrição da metologia utilizada.

O capítulo 4 é das aplicações. Primeiramente, é realizada uma visualização de dados e análise exploratória. Em seguida é feita uma análise de correlação e estatística descritivas e, por fim, uma modelagem preditiva é realizada com foco em modelos sARIMA. Em casos de má aderência do modelo, outros serão testados, a saber: Holt-Winters e Theta.

2 REVISÃO BIBLIOGRÁFICA

2.1 Séries Temporais

O problema de previsão de séries temporais apresenta aplicações em diversas áreas do conhecimento e setores da indústria. De acordo com Kihoro, Otieno e Wafula (2004), os modelos inicialmente utilizam-se de abordagens estatísticas buscando prever o valor de uma variável aleatória x_{i+d} por meio de observações históricas de x, entre os quais o mais importante de tais modelos é o *Autoregressive Integrated Moving Average* (ARIMA). Explica-se tal predominância devido à facilidade de implementação de tais modelos e à difundida metodologia Box-Jenkins (BOX et al., 2016).

Efetivamente os modelos ARIMA apresentam performance satisfatória na modelagem a partir de dados empíricos, afirma Wang et al. (2013). Entretanto, assume-se uma correlação linear nos valores de série temporal. Em si, tal afirmação é forte e não factual, pois com frequência se desconhecesse a natureza e complexidade das relações entre os dados. Zhang (2003) elucida que tal premissa é uma das principais limitações de modelos ARIMA, sendo possível extrair valor de modelos mais flexíveis com relação às suas capabilidades de modelagem.

Uma rede neural artificial ou, em inglês, *artificial neural network* (ANN), constitui-se de uma poderosa ferramenta para representar séries temporais por meio da tentativa não paramétrica de emular o comportamento do cérebro humano, reconhecendo padrões e similaridades nos conjuntos de dados (KIHORO; OTIENO; WAFULA, 2004). Sua grande vantagem está relacionada à flexibilidade de não se fixar antecipadamente a natureza dos parâmetros e serem orientados a dados, isto é, tem pouca dependência do conhecimento *a priori* dos fenômenos (KHASHEI; BIJARI, 2011) ou mesmo "da natureza das relações entre os dados"(WANG et al., 2013).

Naturalmente, diversos autores realizaram estudos comparativos entre as performances ARIMA e ANN. Fishwick (1989), por exemplo, reporta uma performance inferior para redes neurais quando comparada a modelos lineares. Tang, Almeida e Fishwick (1991) explicam que tal constatação pode ser implicação de um conjunto de dados linear e sem muitas perturbações. De todo modo, constatou-se a existência de resultados conflitantes na literatura, levando autores como Khashei e Bijari (2011) a considerar leviano o uso premeditado de redes neurais.

Zhang (2003) propõe uma abordagem híbrida na previsão de séries temporais, tratando a linearidade do modelo através de ARIMA e a não linearidade através de ANN. Com isso, o modelo combinado se torna mais robusto e menos suscetível a possíveis mudanças estruturais causadas por novos dados. Wang et al. (2013) posteriormente avaliam

e confirmam o trabalho de Zhang (2003), concluindo que efetivamente um modelo híbrido ARIMA-ANN é superior a cada um deles individualmente.

É possível notar o crescente interesse de pesquisadores e do setor privado em técnicas de previsão, o que se manifesta de forma patente através de competições em busca de modelos e métodos cuja aplicação a casos reais resultassem nas melhores previsões - as competições de Madridakis. Um método bastante robusto utilizado como referência é o método Theta (ATHANASOPOULOS; HYNDMAN; WU, 2011), cuja simplicidade e robustez atraiu a atenção de pesquisadores já na competição M3 de 2000 (KONING et al., 2005).

Devido ao cenário evidenciado pelos estudos de Ogino et al. (2021) para a região Centro-Oeste, que representa grande parte das operações comerciais e de produção de fertilizantes minerais da Eurochem Fertilizantes Tocantins, entende-se conveniente agregar ferramentas cada vez robustas para suportar a tomada de decisão. Para auxiliar o ciclo do *S&OP*, o objetivo deste trabalho é utilizar ferramentas de aprendizado dinâmico para gerar uma perspectiva temporal do consumo de matérias-primas a ser implementado na tomada de decisão das previsões de produção das unidades. Propõe-se, assim, a utilização de modelos ARIMA para descrição matemática de demanda das matérias-primas, aplicando-o na criação de modelos de previsão. A robustez da capacidade preditiva será avaliada comparativamente, aplicando-se concomitantemente outros métodos, quais sejam: método Theta e Holt-Winters.

2.2 Conceitos Iniciais

2.2.1 Das definições

Brockwell e Davis (1991) oferecem um conjunto de definições importantes para compreensão de séries temporais. Preliminarmente, entende-se um processo estocástico como uma família de variáveis aleatórias $\{X_t, t \in T\}$, definidas em um espaço de probabilidades ω. As funções $\{X(\alpha), \alpha \in \omega\}$ em uma ordem definida de T representam a realização de um processo estocástico. Uma série temporal, portanto, é representada tanto por um conjunto de dados quanto por um processo através do qual esses dados se manifestam. Outro conceito importante derivado de uma definição preconizada pelos autores é a de estacionariedade estrita. Garante-se tal condição se a distribuição de probabilidade conjunta para $\{X_t, ..., X_{t+k}\}$ e $\{X_{t+h}, ..., X_{t+k+h}\}$ for a mesma para todo $h \in Z$. Assegurar a estacionariedade é necessário para a aplicação de técnicas derivadas da teoria de processos estacionários.

Efetivamente, Ehlers (2009) elucidam características particulares de dados referidos como séries temporais, em particular a importância da "ordem temporal das observações", isto é, a trajetória do processo estocástico, tradicionalmente decomposto em 3 componentes: tendência T_t, sazonalidade C_t e componente aleatória R_t. O maior foco desse trabalho se

concentra na elucubração acerca das componentes tendência e sazonalidade.

$$X_t = T_t + C_t + R_t \tag{2.1}$$

A susceptibilidade de um conjunto de dados temporais de se repetir a cada s períodos de tempo pode ser grosseiramente definido como sazonalidade. Tal comportamento pode ser caracterizado por flutuações constantes ou aproximadamente constantes ao longo do tempo, ao que se atribui a denominação de sazonalidade aditiva. Por outro lado, o nível da série pode impactar no tamanho das flutuações. Nesses casos, as séries são ditas com sazonalidade multiplicativa (MORETTING; TOLOI, 2004; EHLERS, 2009).

Tendência indica o sentido de crescimento ou decrescimento da série, conforme explicam Ehlers (2009) e Moretting e Toloi (2004), e pode ser de 3 tipos. O primeiro é o linear, em que a taxa de crescimento (ou decrescimento) é constante. O segundo, por outro lado, essa taxa apresenta um multiplicador para cada ocorrência temporal de sua variável. Por fim, o crescimento amortecido ocorre quando há decrescimento da série.

2.2.2 Operadores

É conveniente definir alguns operadores utilizados para descrição e manipulação matemática das séries temporais. Seja, portanto, a realização de um processo estocástico em T definido através de um conjunto de variáveis aleatórias $\{X_t, t \in T\}$ em um espaço de probabilidades ω. O operador **backshift** ou **retardo** aplicado m vezes a X_t resultará em X_{t-m}, como evidencia a equação 2.2.

$$B^m X_t = X_{t-m} \tag{2.2}$$

De maneira semelhante, o operador **forward** ou **translado** desloca X_t em m vezes até $X_t + m$, conforme a equação 2.3.

$$F^m X_t = X_{t+m} \tag{2.3}$$

É mais usual, entretanto, representar-se as operações matemáticas em função do operador retardo, que tem aplicação mais intuitiva e imediata. Por exemplo, o operador **diferença** aplicado a X_t é a diferença entre a própria variável X_t e seu backshift. Assim:

$$\Delta X_t = X_t - X_{t-1} \tag{2.4}$$

$$\Delta X_t = X_t - BX_t \tag{2.5}$$

$$\Delta X_t = (1 - B)X_t \tag{2.6}$$

O operador **soma** é definido pelo somatório das n observações da variável aleatória, isto é:

$$SX_t = X_t + X_{t-1} + ... + X_{t-n-1} \tag{2.7}$$

$$SX_t = X_t + BX_t + ... + B^{n-1}X_t \tag{2.8}$$

$$SX_t = X_t + BX_t + ... + B^{n-1}X_t = (1 + B + ... + B^{n-1})X_t \tag{2.9}$$

$$SX_t = (1 - B)^{-1}X_t = \Delta^{-1}X_t \tag{2.10}$$

2.2.3 Autocovariância e Autocorrelação

Ao se compreender uma série temporal pela ótica explanada por Brockwell e Davis (1991), em que a variável aleatória X_t é a manifestação da realização de um processo estocástico, depreende-se que existe uma relação implícita entre os valores de X_t e o de seu retardo ou translado. Essa compreensão é central e edifica objetivamente a construção de modelos de séries temporais, pois tomados quaisquer duas tais variáveis no espaço de probabilidades ω, é possível aferir a força de sua relação.

Casella e Berger (2001) discutem e evidenciam que entre duas variáveis aleatórias (y, z) existe a presença de uma relação, que pode ser forte ou fraca. A variância é uma medida da esperança do quadrado da dispersão de uma variável em relação a sua própria média.

$$var(y) = E[(y - \mu_y)^2] \tag{2.11}$$

$$var(z) = E[(z - \mu_z)^2] \tag{2.12}$$

Similarmente, a covariância afere essa dispersão inter-relacional das variáveis. Portanto, pela definição dos autores então a covariância é dada pela equação (2.13). Outra medida que se desdobra do conceito de covariância é a correlação, equação (2.14), que em termos práticos afere a força das relação entre as variáveis, porém, garantindo que $-1 \leq \rho_{yz} \leq 1$ por consequência do teorema da desigualdade de Cauchy-Schwarz.

$$cov(y, z) = \sigma^2_{yz} = E[(y - \mu_y)(z - \mu_z)] \tag{2.13}$$

$$\rho_{yz} = \frac{\sigma_{yz}^2}{\sigma_y \sigma_z} = \frac{E[(y - \mu_y)(z - \mu_z)]}{\sigma_y \sigma_z} \tag{2.14}$$

Suponha-se, então, as variáveis aleatórias $\{X_t, t \in T\}$ consideradas anteriormente. Sejam $y = X_t$ e $z = X_{t+\tau}$, entendem-se as funções autocovariância e autocorrelação como uma medida de dispersão inter-relacional entre manifestações de uma variável aleatória dentro da realização do processo estocástico, portanto, dada pela variância e correlação de X_t em relação a $X_{t+\tau}$, conforme se define em Brockwell e Davis (1991), Moretting e Toloi (2004) e Ehlers (2009). Pode-se, então, computar a autocorrelação de uma série temporal em relação ao seu retardo como ferramenta exploratória de seu comportamento - o correlograma.

$$\gamma(\tau) = cov(X_t, X_{t+\tau}) = E[(X_t - \mu)(X_{t+\tau} - \mu)] \tag{2.15}$$

$$\rho(\tau) = \frac{cov(X_t, X_{t+\tau})}{\sigma_{Xt}\sigma_{Xt+\tau}} = \frac{E[(X_t - \mu)(X_{t+\tau} - \mu)]}{\sigma_{Xt}\sigma_{Xt+\tau}} \tag{2.16}$$

2.2.4 Métricas de Erro

Ao evidenciar que o erro absoluto percentual médio simétrico (sMAPE, do inglês, symmetric mean absolute percentage error) não trata de maneira igualitária erros positivos e negativos, Fiorucci (2016) justifica a propositura por Hyndman e Koehler (2006) da métrica erro médio absolute escalado (MASE, do inglês, *Mean Absolute Scaled Error*). Mesmo considerando essa limitação no tratamento entre erros absolutos positivos e negativos, a métrica de erro que será utilizada nesse trabalho é o Mean Absolute Percentage Error, MAPE, que pode ser representado pela expressão abaixo.

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\frac{|y_t - y_t^{'}|}{|y_t|} \tag{2.17}$$

2.3 O modelo Sazonal Autorregressivo Integrado de Médias Móveis - sARIMA

2.3.1 O caso geral

Conforme descrito por Zhang (2003), em um modelo de médias móveis autoregressivo integrado (ARIMA), o valor futuro de uma variável aleatória é dado como uma função linear de observações passadas e erros randômicos. Desse modo, uma observação futura X_t pode ser descrita pela equação 2.19, onde X_t e ε_t representam, respectivamente, o valor da variável e o erro randômico observado. Ademais, φ_j (j = 1, 2, ..., p) e ϑ_i (i = 0, 1, 2, ..., q) são os parâmetros do modelo; p e q são números inteiros denominados de ordens do modelo; e, por fim, assume-se que o erro randômico tem distribuição idêntica com média

$\mu = 0$ e desvio padrão σ^2. Sua generalização é obtida da equação 2.18 que evidencia os elementos interiores da equação.

$$X_t = \vartheta_0 + \varphi_1 X_{t-1} + \varphi_2 X_{t-2} + ... + \varphi_p X_{t-p} + \varepsilon_t + \vartheta_1 \varepsilon_{t-1} + \vartheta_2 \varepsilon_{t-2} + ... + \vartheta_q \varepsilon_{t-q} \quad (2.18)$$

$$X_t - \varepsilon_t = \sum_{j=1}^{p} \varphi_j X_{t-j} + \sum_{i=0}^{q} i\vartheta_i \varepsilon_{t-i} \quad (2.19)$$

Pela metodologia proposta por Box e Jenkins (1970) e baseada em trabalhos anteriores de Yule (1926), Wang et al. (2013) preconizam 3 etapas iterativas: identificação do modelo, estimação de parâmetros e diagnóstico. No primeiro passo, é necessário realizar uma transformação de dados de modo a garantir que a série temporal seja estacionária, isto é, eliminando efeitos sazonais e tendências e garantindo independêndia para a variável aleatória no eixo temporal. A segunda etapa consiste em estimar os parâmetros e ordens do modelo φ_j (j = 1, 2, ..., p) e ϑ_i (i = 0, 1, 2, ..., q), o que pode ser realizado através de procedimentos de otimização não linear. Por fim, resta verificar se as premissas do modelo acerca do erro, ε_t, são satisfeitas.

A equação 2.19 representa, excluindo-se a componente sazonal, o caso mais geral de um modelo autoregressivo integrado de médias móveis, decomposta em componentes tendência e aleatória. Os casos ainda mais gerais (e frequentemente mais aplicáveis aos fenômenos observáveis) incluem também uma outra componente: sazonalidade.

2.3.2 As componentes do modelo ARIMA

Um modelo auto-regressivo (AR) existe quando é possível afirmar que o valor atual de uma série depende somente do seu passado imediato e um erro aleatório. Ou seja, pode ser descrito através da equação 2.20, que se assemelha a uma regressão múltipla. Quando reescriva em função do operador backshift, aplicado a 2.21, representa-se-la pela equação 2.22.

$$X_t = \vartheta_0 + \varphi_1 X_{t-1} + \varphi_2 X_{t-2} + ... + \varphi_p X_{t-p} + \varepsilon_t \quad (2.20)$$

$$X_t - \vartheta_0 - \varphi_1 X_{t-1} - \varphi_2 X_{t-2} - ... - \varphi_p X_{t-p} = \varepsilon_t \quad (2.21)$$

$$(1 - \varphi_1 B - \varphi_2 B^2 - ... - \varphi_p B^p) X_t = \varepsilon_t \quad (2.22)$$

Por outro lado, em um modelo de médias móveis (MA), não se observa uma correlação entre a manifestação presente de X_t em relação à sua defasagem em qualquer

grau. Desse modo, a série temporal é dita puramente aleatória. Então, descreve-se a série temporal através de 2.23, reduzida à equação 2.24 quando aplicado o operador backshift.

$$X_t = \varepsilon_t + \vartheta_1\varepsilon_{t-1} + \vartheta_2\varepsilon_{t-2} + ... + \vartheta_q\varepsilon_{t-q} \tag{2.23}$$

$$X_t = (1 + \vartheta_1 B + \vartheta_2 B^2 - ... + \vartheta_q B^q)\varepsilon_t \tag{2.24}$$

Nota-se, então, que ao se dizer de uma série X_t em que tanto a compenente aleatória quanto a autoregressiva (ARMA) são evidentes, ambos efeitos se aplicam simultaneamente, ou seja, a equação 2.19 de Zhang (2003) se torna apropriada para representar o processo. Aplicando-se, portanto, o operador backshift, reduz-se o modelo à equação 2.25, igualdade que relaciona os comportamentos auto-regressivos, ao qual designaremos uma função $\phi(B)$, e de médias móveis, $\theta(B)$. Portanto, reduz-se, finalmente, a expressão de um modelo ARMA à 2.26.

$$(1 - \varphi_1 B - \varphi_2 B^2 - ... - \varphi_p B^p)X_t = (1 + \vartheta_1 B + \vartheta_2 B^2 + ... + \vartheta_q B^q)\varepsilon_t \tag{2.25}$$

$$\phi(B)X_t = \theta(B)\varepsilon_t \tag{2.26}$$

Através da trajetória de uma série temporal, é possível ainda que um dado X_t quando deslocado τ até $X_{t+\tau}$ represente um processo estocástico cujo fenômeno que a manifesta seja diferente. Nesse caso, não se pode garantir que características como variância e média se mantenham constantes, portanto, enquanto que para X_t temos um espaço de probabilidades ω, para $X_{t+\tau}$ tem-se um ω' diferente. A estacionariedade da série temporal garante que não haja alteração no espaço amostral. Com muita frequência, não se identifica estacionariedade para conjunto de dados reais, o que pode ser obtido através da aplicação do operador diferença d vezes, produzindo um modelo que pode ser generalizado pela expressão 2.27 e denominado de integrado.

$$\phi(B)\Delta^d X_t = \theta(B)\varepsilon_t \tag{2.27}$$

2.3.3 A sazonalidade

Box e Jenkins (1970), identificando que algumas séries temporais tem a característica de repetir comportamentos a cada s observações, propôs que a série X_t tem forte relação com X_{t-s}, de tal modo que, em se confirmando a presença das componentes auto-regressivas integradas e de médias móveis para a variável X_t, também se pode confirmar a presença de tais componentes quando da defasagem da série em s vezes, o equivalente ao período

de sazonalidade. A forma mais geral da equação que descreve um modelo sazonal autoregressivo integrado de médias móveis é dado pela equação 2.28.

$$\phi(B)\Phi(B^s)\Delta^d\Delta_s^D X_t = \theta(B)\Theta(B^s)\varepsilon_t \tag{2.28}$$

2.4 O método Theta

É uma técnica dinâmica que permite escolher diferentes pesos para as linhas theta θ_1 e θ_2 decompostas a partir da série original e usadas para realizar a previsão, como explica Assimakopoulos e Nikolopoulos (2000), que é aplicado a séries temporais não sazonais ou dessazonalizadas. Fiorucci et al. (2016) descreve o método tradicional (ou standard) através da ótica de referência preconizada por Assimakopoulos e Nikolopoulos (2000), baseada em 5 etapas.

1. Dessazonalização
2. Decomposição
3. Extrapolação
4. Combinação
5. Ressazonalização

Considerando a mesma série temporal X_t definida em ω, Assimakopoulos e Nikolopoulos (2000) propõe que a linha theta é dada pela solução da equação, com $t = 3, ..., n$. Note-se o subscrito l para diferenciar a linha theta do parâmetro θ da componente de médias móveis presente no modelo ARIMA.

$$\Delta^2 Z_t(\theta_l) = \theta_l \Delta^2 X_t \tag{2.29}$$

A solução analítica para a equação 2.29 foi encontrada por Hyndman e Billah (2003), que descrevem as linhas theta como funções obtidas através da regressão linear aplicada diretamente aos dados. Encontra-se, de acordo com os autores, a expressão 2.30, para a qual se encontra duas funções, A_t e B_t, dependentes apenas de X_t e, portanto, não são parâmetros.

$$Z_t(\theta_l) = \theta_l X_t + (1 - \theta_l)(A_n + B_n t) \tag{2.30}$$

$$A_n = \frac{1}{n}\left(\sum_{t=1}^{n} X_t - \frac{n+1}{2} B_n\right) \tag{2.31}$$

$$B_n = \frac{6}{n^2 - 1}\left(\frac{2}{n}\sum_{t=1}^{n} tX_t - \frac{1+n}{n}\sum_{t=1}^{n} X_t\right) \tag{2.32}$$

Para a reconstrução da série original e posterior tarefa de predição, a série original considera pesos iguais para θ_1 e θ_2. Fiorucci et al. (2016) elegantemente descreve uma otimização para esse método, em que os pesos não podem agora assumir diferentes valores, conforme a premissa da equação 2.33.

$$X_t = \omega Z_t(\theta_{l1}) + (1 - \omega)Z_t(\theta_{l2}) \tag{2.33}$$

2.5 Abordagem de redes neurais artificiais aplicados a séries temporais

Em uma rede neural, existe uma camada de *input*, uma ou múltiplas camadas ocultas e a saída de dados, *output*. Khashei e Bijari (2011) descrevem uma modelagem não linear para um conjunto de dados através da equação 2.34, ou seja, descrevem a relação entre entradas e saídas.

$$y_t = w_0 + \sum_{j=1}^{Q} w_j g\left(w_{0j} + \sum_{i=1}^{P} w_{ij} y_{t-i}\right) + \varepsilon_t \tag{2.34}$$

onde $w_{i,j}$ (i = 0, 1, 2,..., P; j = 1, 2,..., Q) e w_j (j = 0, 1, 2,...,Q) são os parâmetros do modelo, P representa o número de nós de entrada na estrutura de rede neural, enquanto Q está para a quantidades de nós escondidos. O modelo se torna, portanto, uma função das observações históricas e de um vetor W, que é o vetor de parâmetros do modelo.

Wang et al. (2013) e Júnior, Oliveira e Neto (2019) corroboram essa abordagem, porém, descrevem o modelo através da equação 2.35.

$$y_t = \alpha_0 + \sum_{j=1}^{Q} \alpha_j g\left(\beta_{0j} + \sum_{i=1}^{P} \beta_{ij} y_{t-i}\right) + \varepsilon_t \tag{2.35}$$

onde $\beta_{i,j}$ (i = 0, 1, 2,..., P; j = 1, 2,..., Q) e α_j (j = 0, 1, 2,...,Q) são os parâmetros do modelo. Novamente, os nós de entrada são representados por P; Q, por sua vez, refere-se à quantidade de nós implícitos. A camada intermediária é representada pela função de transferência logística, a saber:

$$g(x) = \frac{1}{1 + \exp(-x)} \tag{2.36}$$

Em resumo, pode-se descrever o modelo como uma função f em que futuras observações da variável aleatória podem ser descritos por meio de observações históricas e dos pararâmetros definidos nas equações 2.34 ou 2.35.

$$y_t = f\left(y_{t-1}, ..., y_{t-P}, W\right) + \varepsilon_t \tag{2.37}$$

2.6 Modelo Híbrido

Uma avaliação experimental aplicada por Júnior, Oliveira e Neto (2019) a séries temporais propõe e elucida as fases de treinamento e teste de um algoritmo cujo objetivo é aplicar um modelo híbrido entre ARIMA e aprendizado de máquina. Para tal, considera-se a ideia de que o modelo de previsão é composto por duas componetes: linear e não linear. Em particular, considera-se um modelo aditivo. Desse modo, tem-se uma expressão geral representada pela equação 2.38.

$$y_t = Z_t = f(L_t, N_t) = L_t + N_t \tag{2.38}$$

A previsão linear para a série temporal pode ser representada por $L_t^{'}$, de modo que o erro E_t é dado pela diferenteça entre a série original e o valor predito, portanto:

$$E_t = Z_t + L_t^{'} \tag{2.39}$$

Através dos inputs da série de erro, e_t, é possível realizar uma previsão residual e determinar a parte não linear do modelo, $N_t^{'}$. Assim, tem-se uma função dos erros e um erro residual que não pode ser previsto (equação 2.40).

$$N_t^{'} = f(e_{t-1}, e_{t-2}, ..., e_{t-n}) + \varepsilon_t \tag{2.40}$$

Desse modo, de maneira mais geral, a equação 2.41 pode representar o modelo híbrido. Em específico, introduzindo-se a expressão geral para um modelo ARIMA (parte linear) e a equação 2.40 (parte não linear), consegue-se a equação 2.42.

$$Z_t^{'} = L_t^{'} + N_t^{'} \tag{2.41}$$

$$Z_t^{'} = \left(\sum_{j=1}^{p} \varphi_j y_{t-j} - \sum_{i=0}^{q} i\theta_i \varepsilon_{t-i}\right) + f(e_{t-1}, e_{t-2}, ..., e_{t-n}) + \varepsilon_t \tag{2.42}$$

em que φ_j (j = 1, 2, ..., p) e θ_i (i = 0, 1, 2, ..., q) são parãmetros do modelo linear e n representa o tamanho da janela temporal.

3 METODOLOGIA

Este capítulo será dedicado a elucidar as etapas de tratamento de dados necessários para disponibilização de bases históricas estruturadas de faturamento e de consumo de matéria-prima, objetos de interesse desse trabalho. A primeira etapa consiste no esforço de consolidar as informações em relatórios de faturamento e de produção através de consultas ao bancos transacionais do ERP Oracle E-Businesss Suite utilizando SQL. Em seguida, um pré-tratamento é realizado para consolidar um arquivo do tipo CSV que será a entrada de dados dos modelos. Ajustes finos e tratamentos finais são realizados em Python utilizando a biblioteca Pandas para que seja possível realizar as Análises Exploratórias e posteriores testes.

3.1 Consultas aos bancos transacionais usando SQL

Na estrutura de ERP da Eurochem Fertilizantes Tocantins, os dados transacionais de interesse, isto é, volumes faturados para cliente e os consumos de matéria-prima estão disponíveis nos módulos de Vendas, Faturamento, Contas a receber, Produção e Engenharia de Produto. Desse modo, torna-se necessário criar uma estrutura de relatório capaz de consolidar as informações pertinentes ao negócio e disponibilizar através de consulta para o uso das áreas funcionais. Para o propósito específico deste trabalho, duas consultas são de interesse: faturamento e produção.

A primeira consulta, relatório de faturamento, é estruturado de modo a identificar as notas fiscais faturadas para clientes em cada unidade produtiva da ECFTO. Existe um número único que identifica as transações de faturamento para clientes, denominado CUSTOMER_TRX_ID. A partir dessa informação, a consulta é construída buscando informações relacionadas à operação, quais sejam: unidade de faturamento, regional de vendas, data de efetivação da transação, entre outros. O anexo A esboça em sua totalidade o SQL utilizado para criar a consulta de faturamento.

A segunda consulta é complementar ao faturamento, porém, seu objetivo é evidenciar quais foram os lotes expedidos para cliente e quais os volumes de matéria-prima foram consumidos na produção de tais lotes. Por sua natureza, utilizou-se a query de faturamento conforme anexo A e subqueries para consulta aos módulos de produção, identificando o *ID* das transações de produção para buscar os consumos de matéria-prima. Desse modo, o campo CUSTOMER_TRX_ID continua sendo central, porém, torna-se necessário fazer a correlação entre tal campo e TRANSACTION_SOURCE_ID, que identifica a transação de produção do produto acabado. O anexo B elucida o SQL utilizado para gerar a consulta de produção.

3.2 Pré-tratamento de bases de dados usando M

Nativamente, o ERP Oracle E-Business Suite posui conexão com a plataforma Oracle Analytics, que pode ser considerado um serviço de *Business Inteligence* (BI), porém, com motores para *Online Analytical Processing* (OLAP) integrados. Portanto, do ponto de vista de negócio, ambos os relatórios de faturamento e de produção produzem consultas que são disponibilizadas para a plataforma de BI de forma online. Desse modo, foi possível construir dashboards gerenciais com objetivo de acompanhar o faturamento e a produção e consumo de matéria-prima.

Para além da informação gerencial, as consultas geradas também podem ser exportadas no formato CSV através do Oracle Analytics, o que permite realizar um recorte temporal estruturado para ambos relatórios. Esses arquivos, por sua vez, passam por pré-tratamento utilizando recursos da linguagem M através do Power Query.

Em relação à base de faturamento, a consulta do anexo A é executada e exportada para um arquivo CSV através do Oracle Analytics. Então, utilizando o Power Query, é realizada uma leitura desse arquivo. O pré-tratamento consiste em eliminar colunas desnecessárias para análise das séries temporais de entregas de faturamento, produzindo uam estrutura que contenha informações de: data, unidade de faturamento, unidade de produção, regional de vendas e volume transacionado. O anexo C elucida as etapas executadas em M para produzir a estrutura de dados que formará o DataFrame de entregas.

De maneira análoga, a base de produção é obtida através da exportação da consulta do anexo B, também utilizando o Oracle Analytics e gerando um arquivo CSV. Neste caso, o pré-tratamento produz uma estrutura que contenha informações de: data, unidade de faturamento, unidade de produção, código da matéria-prima, descrição da matéria-prima, descrição geral do grupo ao qual a matéria-prima pertence e quantidade consumida. O anexo D evidencia as etapas que produzem a estrutura de dados responsável pelo DataFrame de produção.

3.3 Criação de DataFrames em Pandas

As bases de dados utilizadas para os estudos exploratórios e previsão de séries temporais são duas: *CSV_deliveries.csv* e *CSV_production.csv*. Estes arquivos são armazenados em uma pasta compartilhada do Google Drive, destinada especificamente a criar um repositório de arquivos. A execução, por sua vez, é realizada utilizando notebooks em Python através da plataforma colaborativa Google Colab. São criadas 3 strings, cujos objetivos, em sequência, são: indicar o caminho até o repositório de dados, nomear os arquivos *CSV_deliveries.csv* e *CSV_production.csv*. As instruções abaixo demonstram os comandos executados.

```
from google.colab import drive
drive.mount('/content/drive')
# Vinculando endereços de drive para query
path = '/content/drive/My Drive/USP/TCC/Database/'
file1 = 'CSV_deliveries.csv'
file2 = 'CSV_production.csv'
```

3.3.1 Bibliotecas e Pacotes

Uma vez que o escopo do trabalho é tratar, visualizar e realizar testes de predição em relação às bases que serão separadas diária e mensalmente para cada unidade fabril. Para realizar o tratamento das bases de dados, utilizam-se as bibliotecas do *Pandas* e *Numpy*, que possuem funções e métodos úteis ao trabalhar DataFrames extensos. A visualização de dados é realizada através de métodos da biblioteca do *matplotlib* e *seaborn*.

```
# Importanto as bibliotecas
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

A análise exploratória depende, para além de realizar a visualização dos dados, exibir estatísticas descritivas em relação à série temporal evidenciando o seu comportamento. Com isso, é possível fazer estudos qualitativos e quantitativos que corroboram analiticamente a escolha de modelos preditivos. Do *statsmodels*, importamos as ferramentas de estatística de Dickey-Fuller, *adfuller*. Para decompor a série temporal em suas componentes tendência, sazonalidade e médias móveis, importamos também *seasonal_decompose*.

Para criação dos modelos em si, importa-se do *statsmodels* o *ExponentialSmoothing* para os testes pelo método de Holt-Winters; o *ThetaModel* para testes pelo método Theta; *SARIMAX* para os testes para os modelos do tipo ARIMA. Além disso, do *pmdarima*, importamos o algoritmo de otimização *auto_arima*, que permite encontrar o melhor modelo para fit dos dados. Por fim, para realizar as medidas de erro, importamos o *mean_absolute_percentage_error*.

```
# Importando pacotes necessários
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_absolute_percentage_error
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.forecasting.theta import ThetaModel
from statsmodels.tsa.holtwinters import ExponentialSmoothing
```

```
from matplotlib import pyplot
from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from pmdarima import auto_arima
%matplotlib inline
```

3.3.2 Criação dos DataFrames

Cria-se um DataFrame global para entregas na granularidade temporal de dias, ao qual se denomina *df1*. Agrupam-se os volumes de entregas em um DataFrame global de entregas mensais chamado de *dm1*. De maneira similar, existe para o relatório de produção os DataFrames globais de consumo de matéria-prima *df2* e *dm2*.

```
## Importação da base de dados de entregas (CSV_deliveries.csv) do ORACLE
df1 = pd.read_csv(path+file1)
df1['Deliveries'].fillna(0,inplace=True)

## Dataframe global
df1['Date'] = pd.to_datetime(df1['Date'])
dm1 = df1.groupby(pd.Grouper(key='Date', freq='M'))['Deliveries'].sum()
#dm.index = dm.index.strftime('%B')
dm1 = pd.DataFrame(dm1)

## Importação da base de dados de entregas (CSV_production.csv) do ORACLE
df2 = pd.read_csv(path+file2)
df2['Quantity'].fillna(0,inplace=True)

## Dataframe global
df2['Date'] = pd.to_datetime(df2['Date'])
dm2 = df2.groupby(pd.Grouper(key='Date', freq='M'))['Quantity'].sum()
#dm2.index = dm2.index.strftime('%B')
dm2 = pd.DataFrame(dm2)
```

Para atender os objetivos de negócio, criam-se também DataFrames equivalentes aos globais, porém, com slicing para cada unidade produtiva da empresa, criando uma estrutura única para cada fábrica. Desse modo, é possível realizar análises exploratórias e estudos preditivos para unidades individualmente. O notebook completo dos tratamentos de dados executados, bem como os testes executados estão disponíveis no anexo E.

Uma vez que para o escopo desse projeto trabalhamos apenas com a unidade de São Luís, elucida-se a execução das etapas de transformação de dados para recuperar informações históricas para para a unidade de código SLO do DataFrame de entregas original *df1*. Desse modo, inicialmente, é realizado um slicing para essa unidade, seguido por uma cópia que será destinada à base mensal. Em ambos os casos, utiliza-se o método *groupby* do Pandas para realizar os agrupamentos da soma dos volumes de entrega, *Deliveries*, em função das datas, sendo a frequência diáras para *df1* e mensal para *dm1*.

```
# Criando dataframes para São Luís

df1_SLO = df1[(df1['pUnit'] == 'SLO')]

dm1_SLO = df1_SLO.copy()
dm1_SLO['Date'] = pd.to_datetime(df1_SLO['Date'])
dm1_SLO = df1_SLO.groupby(pd.Grouper(key='Date',
                                     freq='M'))['Deliveries'].sum()
#dm.index = dm.index.strftime('%B')
dm1_SLO = pd.DataFrame(dm1_SLO)
```

Para os casos em que existem valores nulos ou negativos na base diária, é necessário tratá-los, uma vez que podem afetar os resultados na fase de testes. Desse modo, cria-se uma máscara booleana responsável por filtrar os valores nulos e, posteriormente, aplicar ao DataFrame. Isso garante que todos os dias do calendário sejam considerados, inclusive aqueles que não tiveram faturamento ou este valor foi negativo por conta de devoluções.

```
df1_SLO = df1_SLO.groupby(by='Date',as_index=False)['Deliveries'].sum()
mask_SLO = df1[(df1['pUnit'].isnull())]
                    .drop(['bUnit','pUnit','Regional'],axis='columns')
df1_SLO = df1_SLO.append(mask_SLO)
df1_SLO['Date'] = pd.to_datetime(df1_SLO['Date'])
df1_SLO = df1_SLO.set_index('Date').sort_index(axis = 0)

idx = pd.date_range(start = df1_SLO.index.min(),
                    end = df1_SLO.index.max(),
                    freq='D')
df1_SLO = df1_SLO.reindex(idx)

df1_SLO[df1_SLO['Deliveries'] < 0] = 0
df1_SLO = df1_SLO.fillna(0)
```

De maneira semelhante, para o caso do consumo de matérias-primas, também precisamos iniciar com um slicing em *df2* para filtrar apenas dados referentes a São Luís. Além disso, também é necessário filtrar e armazenar em diferentes DataFrames o equivalente de consumo para cada uma das 5 matérias-primas selecionadas: MP1, cloreto de potássio; MP2, ureia; MP3, MAP; MP4, TSP; e MP5, SSP. Após esses passos, as informações são novamente agrupadas utilizando o *groupby* do Pandas pela soma das quantidades, *Quantity*, de cada matéria-prima. O processo é realizado criando-se DataFrames de frequência diária e mensal.

```
# Criando DataFrames para São Luís
df2_SLO = df2[(df2['pUnit'] == 'SLO')]
df2_SLO_MP1 = df2_SLO[(df2_SLO['ProductGenDesc'] == MP1)]
df2_SLO_MP2 = df2_SLO[(df2_SLO['ProductGenDesc'] == MP2)]
df2_SLO_MP3 = df2_SLO[(df2_SLO['ProductGenDesc'] == MP3)]
df2_SLO_MP4 = df2_SLO[(df2_SLO['ProductGenDesc'] == MP4)]
df2_SLO_MP5 = df2_SLO[(df2_SLO['ProductGenDesc'] == MP5)]

# Para MP1 (Cloreto de Potássio)
dm2_SLO_MP1 = df2_SLO_MP1.copy()
dm2_SLO_MP1 = dm2_SLO_MP1.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_SLO_MP1 = pd.DataFrame(dm2_SLO_MP1)
df2_SLO_MP1 = df2_SLO_MP1.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_SLO_MP1 = pd.DataFrame(df2_SLO_MP1)

#Para MP2 (Ureia)
dm2_SLO_MP2 = df2_SLO_MP2.copy()
dm2_SLO_MP2 = dm2_SLO_MP2.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_SLO_MP2 = pd.DataFrame(dm2_SLO_MP2)
df2_SLO_MP2 = df2_SLO_MP2.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_SLO_MP2 = pd.DataFrame(df2_SLO_MP2)

#Para MP3 (MAP)
dm2_SLO_MP3 = df2_SLO_MP3.copy()
dm2_SLO_MP3 = dm2_SLO_MP3.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
```

```
dm2_SLO_MP3 = pd.DataFrame(dm2_SLO_MP3)
df2_SLO_MP3 = df2_SLO_MP3.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_SLO_MP3 = pd.DataFrame(df2_SLO_MP3)

#Para MP4 (TSP)
dm2_SLO_MP4 = df2_SLO_MP4.copy()
dm2_SLO_MP4 = dm2_SLO_MP4.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_SLO_MP4 = pd.DataFrame(dm2_SLO_MP4)
df2_SLO_MP4 = df2_SLO_MP4.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_SLO_MP4 = pd.DataFrame(df2_SLO_MP4)

#Para MP5 (SSP)
dm2_SLO_MP5 = df2_SLO_MP5.copy()
dm2_SLO_MP5 = dm2_SLO_MP5.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_SLO_MP5 = pd.DataFrame(dm2_SLO_MP5)
df2_SLO_MP5 = df2_SLO_MP5.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_SLO_MP5 = pd.DataFrame(df2_SLO_MP5)
```

3.4 Requisitos e Programação para Análise Exploratória

A etapa de análise exploratória é essencial para a compreensão do comportamento das séries temporais. Em específico, relacionado ao dados que fazem parte do escopo desse trabalho, é necessário entender que cada DataFrame terá um comportamento particular e diferentes tendências, sazonalidades e médias móveis. Portanto, para realizar a análise exploratória, foram alguns passos, a saber:

- Análise gráfica
- Separação em base treino e teste em diferentes DataFrames;
- Aplicar *seasonal_decompose* para avaliar as componentes da série temporal
- Exibir a estatística de Dickey-Fuller para a série temporal
- Realizar otimização stepwise utilizando o *auto_arima* para determinar melhor modelo ARIMA

3.4.1 Visualização de dados

Os volumes de entrega diários de São Luís na base faturamento são obtidos através do DataFrame *df1_SLO* em que o eixo x são os índices de data e o eixo y os volumes transacionados. Do Seaborn, utiliza-se o método *lineplot*, indicando o índice como eixo x e a coluna *Deliveries* como eixo y. Similarmente, para a base de faturamento mensal, usamos o DataFrame *dm1_SLO*.

```
#Base diária
dSLO = sns.lineplot(x=df1_SLO.index[:], y = df1_SLO['Deliveries'][:],
                    color = 'blue', dashes=df1_SLO['Deliveries'])
dSLO.set_title('São Luís - Entregas Diárias (toneladas)', fontsize = 15)
dSLO.set_ylabel('Entregas (ton)', fontsize = 13)
dSLO.set_xlabel('Dias', fontsize = 13)
plt.show()

#Base mensal
mSLO = sns.lineplot(x=dm1_SLO.index[:], y = dm1_SLO['Deliveries'][:],
                    color = 'red', dashes=dm1_SLO['Deliveries'])
mSLO.set_title('São Luís - Entregas Mensais (toneladas)', fontsize = 15)
mSLO.set_ylabel('Entregas (ton)', fontsize = 13)
mSLO.set_xlabel('Dias', fontsize = 13)
plt.show()
```

Similarmente, para os gráficos de consumo de matéria-prima, utilizam-se os DataFrames de consumo mensal de matéria-prima para cada uma dos 5 produtos que serão acompanhados: cloreto de potássio, ureia, MAP, TSP e SSP. Para o cloreto, por exemplo, utiliza-se o *lineplot* do Seaborn, com eixo x como índice do DataFrame e eixo y com a coluna de quantidade de consumo de matéria-prima, *Quantity*. Repete-se a estrutura de comando para exibir gráficos para os demais produtos, substituindo *MP1* por *MP2, MP3, MP4* ou *MP5*.

```
#Cloreto de Potássio
mSLO_MP1 = sns.lineplot(x=dm2_SLO_MP1.index[:],
                        y = dm2_SLO_MP1['Quantity'][:],
                        color = 'red',
                        dashes=dm2_SLO_MP1['Quantity'])
title = 'São Luís - Entregas Mensais de Cloreto de Potássio (toneladas)'
mSLO_MP1.set_title(title, fontsize = 15)
mSLO_MP1.set_ylabel('Entregas (ton)', fontsize = 13)
```

```
mSLO_MP1.set_xlabel('Dias', fontsize = 13)
plt.show()
```

3.4.2 Separação em bases de treino e teste

Para viabilizar o treinamento de modelos de previsão de demanda, separa-se as bases de faturamento e produção em teste e treino de acordo com suas características específicas, de modo a capturar melhor as componentes das séries temporais. Para as entregas diárias, considera-se como base de treino todo o registro histórico até os últimos 20 dias do calendário. Os 7 dias seguintes são base de teste. Para as bases mensais de entregas, por outro lado, consideram-se toda base histórica até os último 5 meses como treino, o restante como teste. Por fim, para as bases mensais de consumo de matéria-prima, o recorte temporal ocorre nos últimos 4 meses para cloreto e nos últimos 3 meses para as demais.

```
treino_d1SLO = df1_SLO[:-20]
teste_d1SLO = df1_SLO[-20:-13]
treino_m1SLO = dm1_SLO[:-5]
teste_m1SLO = dm1_SLO[-5:]

treino_m2SLO_MP1 = dm2_SLO_MP1[:-4]
teste_m2SLO_MP1 = dm2_SLO_MP1[-4:]
treino_d2SLO_MP1 = df2_SLO_MP1[:-25]
teste_d2SLO_MP1 = df2_SLO_MP1[-25:-18]

treino_m2SLO_MP2 = dm2_SLO_MP2[:-3]
teste_m2SLO_MP2 = dm2_SLO_MP2[-3:]
treino_d2SLO_MP2 = df2_SLO_MP2[:-25]
teste_d2SLO_MP2 = df2_SLO_MP2[-25:-18]

treino_m2SLO_MP3 = dm2_SLO_MP3[:-3]
teste_m2SLO_MP3 = dm2_SLO_MP3[-3:]
treino_d2SLO_MP3 = df2_SLO_MP3[:-25]
teste_d2SLO_MP3 = df2_SLO_MP3[-25:-18]

treino_m2SLO_MP4 = dm2_SLO_MP4[:-3]
teste_m2SLO_MP4 = dm2_SLO_MP4[-3:]
treino_d2SLO_MP4 = df2_SLO_MP4[:-25]
teste_d2SLO_MP4 = df2_SLO_MP4[-25:-18]
```

```
treino_m2SLO_MP5 = dm2_SLO_MP5[:-3]
teste_m2SLO_MP5 = dm2_SLO_MP5[-3:]
treino_d2SLO_MP5 = df2_SLO_MP5[:-25]
teste_d2SLO_MP5 = df2_SLO_MP5[-25:-18]
```

3.4.3 Viabilidade dos Modelos de Previsão

A avaliação das séries temporais é executada através de sua decomposição em componentes tendência, sazonalidade e componente aleatória utilizando o *seasonal_decompose*. Além disso, exibe-se as estatística de Dickey-Fuller através do *adfuller*, o que permite aceitar ou rejeitar a hipótese de estacionariedade, condição necessária para modelagem. Por fim, os correlogramas são exibidos, de modo que seja possível inferir com grau de significância a autocorrelação entre o valor da variável aleatória ao longo da trajetória do processo estocástico. Para isso, utilizam-se o *plot_acf* e *plot_pacf*. A seguir, evidencia-se a execução dos testes para o caso do DataFrame *df1_SLO*. A busca pelo melhor modelo ocorre pela minimização do AIC com o algoritmo de otimização *auto_arima*.

```
# Seasonal Decompose das Entregas Diárias
result_deliveriesd_SLO = seasonal_decompose(treino_d1SLO['Deliveries'],
                                            model='additive', freq=7)

result_deliveriesd_SLO.plot()
pyplot.show()

result_m = adfuller(df1_SLO['Deliveries'], autolag='AIC')
print(f'Estatítica de Dickey-Fuller: {result_m[0]} \n')
print(f'p-Valor: {result_m[1]}')
for key, value in result_m[4].items():
  print('\t%s: %.3f' % (key,value))

lags = 120
title1 = 'Autocorrelação: Entregas Diárias em São Luís'
plot_acf(df1_SLO['Deliveries'],title=title1,lags=lags);
title2 = 'Autocorrelação Parcial: Entregas Diárias em São Luís'
plot_pacf(df1_SLO['Deliveries'],title=title2,lags=lags);
```

4 MODELAGEM E RESULTADOS

Este capítulo é dedicado aos resultados e desenvolvimentos inerentes ao processo de criação de modelos preditivos das séries temporais de entregas e de consumo de matéria-prima. A estrutura de análise é semelhante, porém, individualizada, pois cada série temporal tem características próprias. Portanto, as seções serão dedicadas a cada DataFrame estudado, com suas subseções dedicadas, respectivamente, à análise exploratória com visualização de dados e decomposição da série temporal, além da análise de correlações, estatísticas descritivas, obtenção de um modelo de previsão e avaliação de erros.

4.1 São Luís: Volume de Entregas Diárias

4.1.1 Visualização de Dados

A observação dos dados históricos sugere que existem dois tipos de sazonalidade: anual e semanal. A observação anual é factual, haja visto que existem dois grandes ciclos de entregas de fertilizantes nas fábricas ao longo do ano: safra e safrinha, separados por um período conhecido como entressafra, normalmente no mês de abril, conforme ilustrado pela Figura 1.

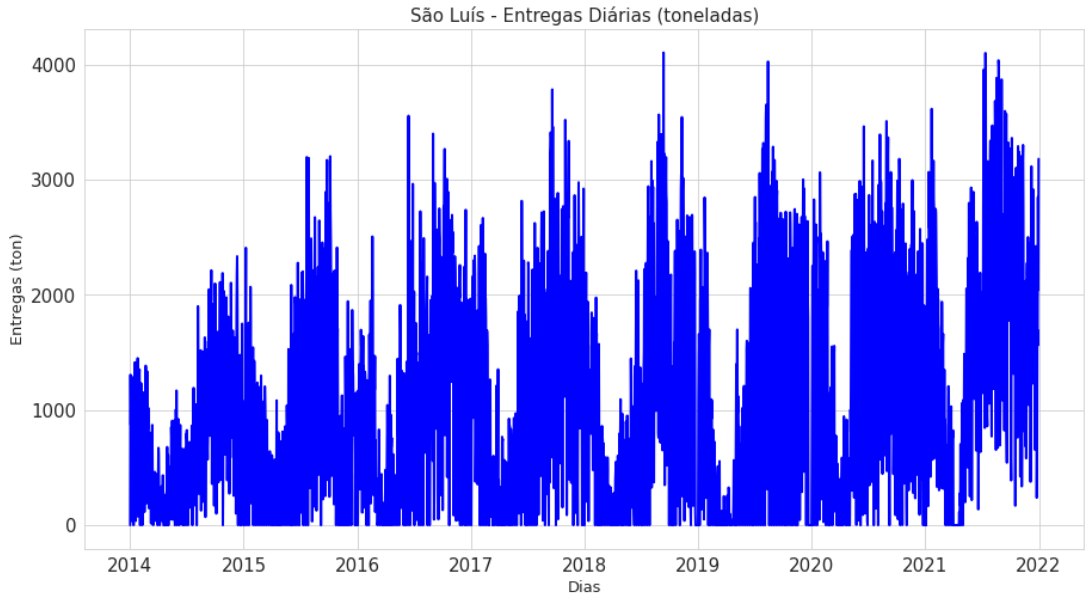

Figura 1 – Entregas diárias em São Luís (toneladas)

Fonte: Elaborado pelo autor

Além disso, a sazonalidade semanal se deve aos turnos operacionais das fábricas, uma vez que são raros os turnos aos domingos. Pela Figura 2, fica clara a existência de uma sazonalidade semanal, mostrando inclusive que existe um vale de entregas no mês de

Abril. Não se depreende objetivamente uma componente tendência, portanto, para fins de modelagem, supõe-se que seja do tipo aditiva.

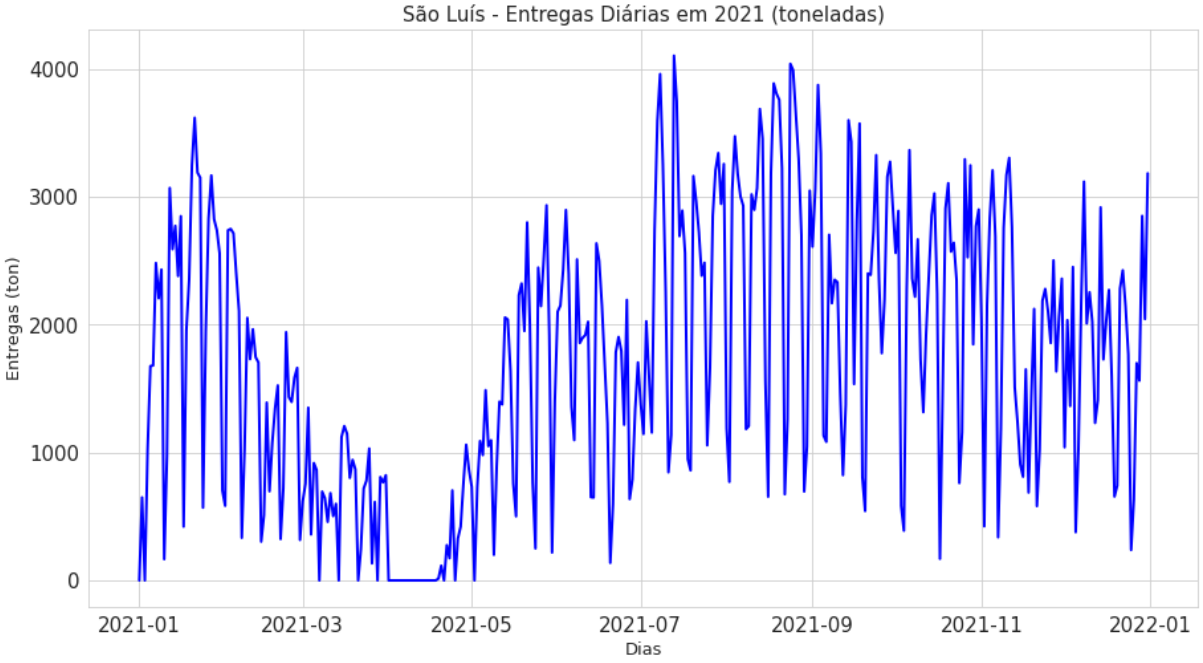

Figura 2 – Entregas diárias em São Luís no ano de 2021 (toneladas)

Fonte: Elaborado pelo autor

Efetivamente, ao se realizar a decomposição da série temporal em suas componentes, conforme a Figura 3, observa-se que a componente de tendência evidencia aumento no volume de entregas no mês de janeiro, pico da safrinha, e após o mês de junho, quando se inicia o caminho para a safra. A componente sazonal, neste caso na granularidade de semana, é evidente e corrobora a ideia inicial acerca dos turnos de fábrica. Além disso, existe um proeminente fator residual aleatório.

4.1.2 Análise de Correlação e Estatísticas Descritivas

A análise dos correlogramas da Figuras 4 e 5 sugerem que, considerado um espaço de probabilidades da variável aleatória de entregas diárias em São Luís, existe forte correlação entre seu valor e as defasagem até um $lag = 40$, aproximadamente. Mais do que isso, há indícios de que os dias produtivos durante a semana correlacionam-se fortemente com seus equivalentes em semanas anteriores, isto é, as entregas de uma segunda podem ser comparadas às da segunda da semana anterior.

Um teste de Dickey-Fuller resulta em um **p-valor = 0,001123**, consequentemente, aceitando a hipótese de estacionariedade da série temporal. Efetivamente, como foi comprovado na análise exploratória, a componente tendência não indica um sentido de crescimento ou decrescimento.

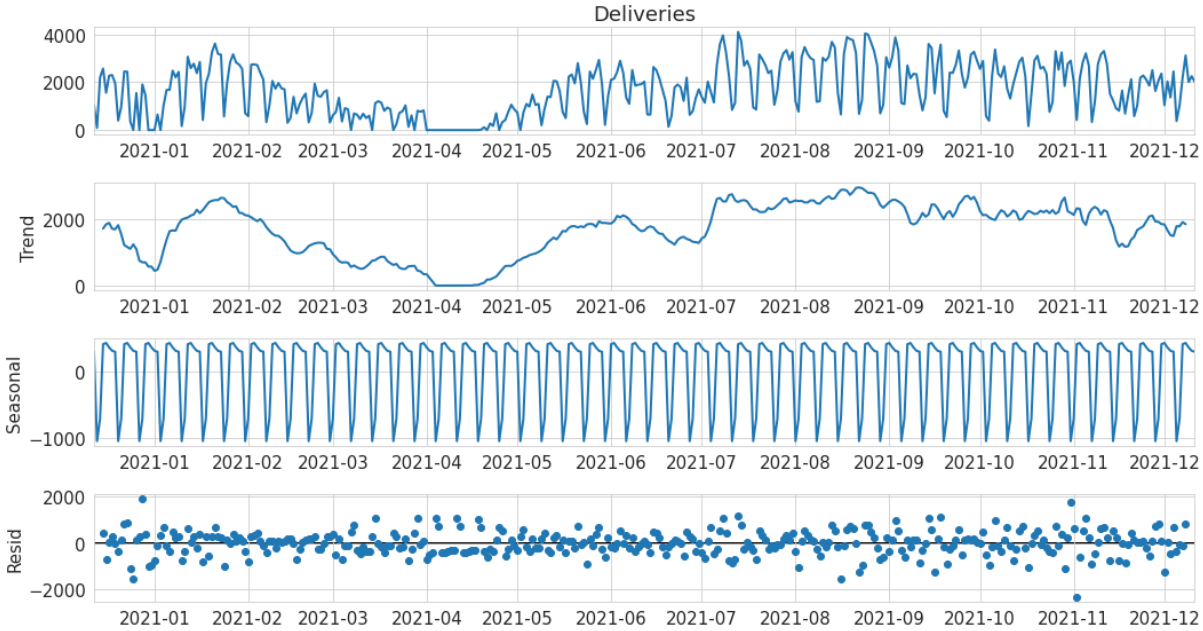

Figura 3 – Decomposição das componentes da série temporal de entregas diárias

Fonte: Elaborado pelo autor

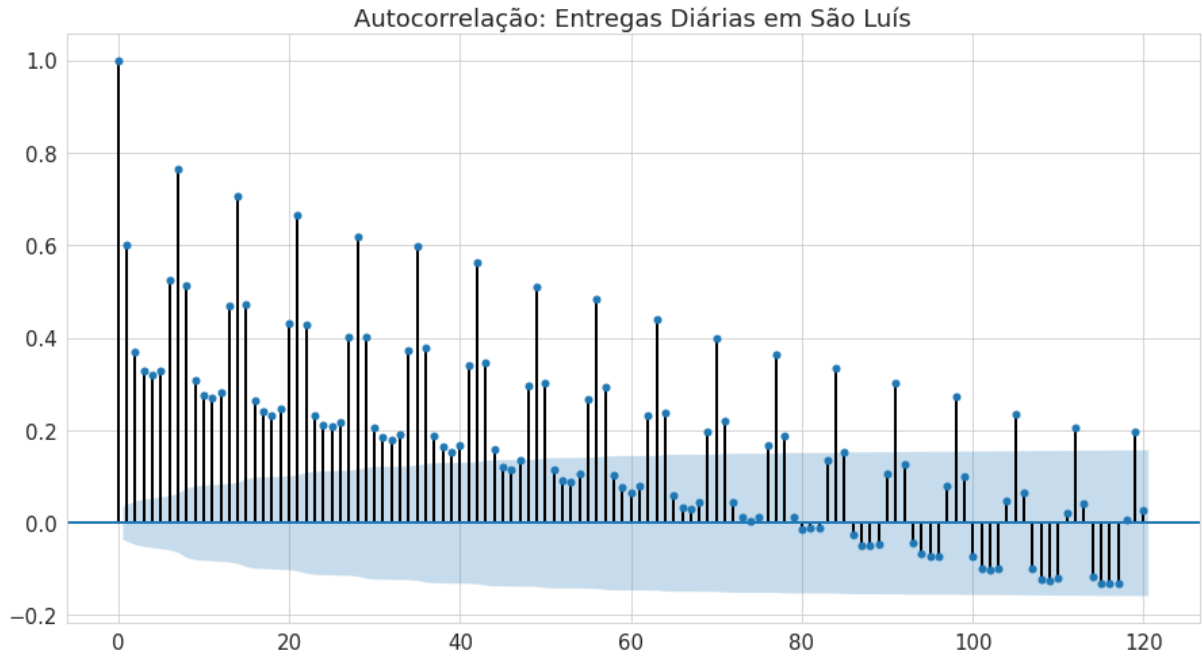

Figura 4 – Análise de Autocorrelação para Entregas Diárias

Fonte: Elaborado pelo autor

4.1.3 Modelagem Preditiva e Forecast

Utilizando o algoritmo de otimização *auto_arima*, busca-se pelo modelo que minimiza o AIC e revela a melhor série temporal que modela o processo estocástico da variável

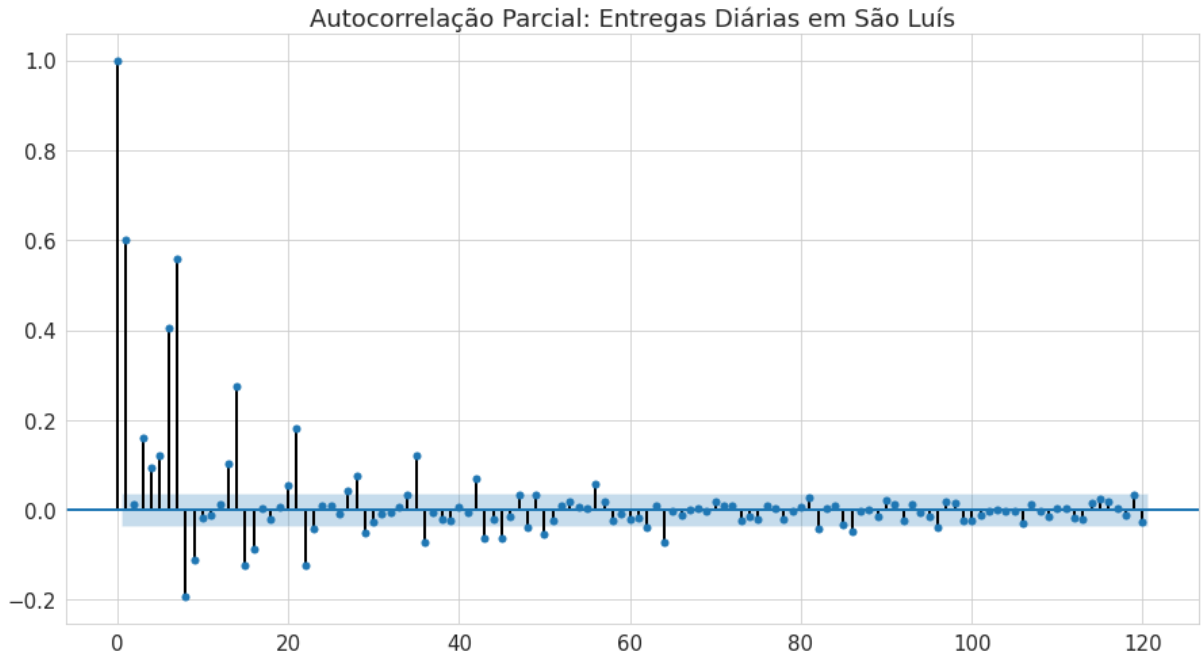

Figura 5 – Análise de Autocorrelação Parcial para Entregas Diárias

Fonte: Elaborado pelo autor

aleatória, representado pela equação 2.28. Para o caso de entregas diárias de São Luís, observa-se que o modelo ótimo é representado por um $\mathbf{sARIMA(6,1,0)(1,0,2)[7]}$. A Figura 6 representa o fit que estima os parâmetros da equação 2.28. Neste caso, o MAPE aferido entre valores preditos e teste é de MAPE $= 22,99\%$. A Figura 7, por fim, representa um forecast de 7 dias a frente.

4.2 São Luís: Volume de Entregas Mensais

4.2.1 Visualização de Dados

Em complemento à elucidação acerca do comportamento de entregas realizado na seção 4.1.1, a Figura 8 evidencia ainda mais claramente os períodos de entressafra presentes nos meses de abril e março. Há, portanto, evidências claras de sazonalidade anual, isto é, safra e safrinha. A componente tendência fica claramente evidenciada ao longo da realização do processo estocástico, mostrando o crescimento histórico na produtividade anual. Supõe-se que a tendência tenha modelo aditivo.

4.2.2 Análise de Correlação e Estatísticas Descritivas

A decomposição da série temporal de entregas mensais revela, efetivamente, que a componente tendência é crescente ao longo dos anos, porém, com clara delimitação entre dois diferentes períodos do ano, conforme mostra a Figura 9. Isso corrobora as conclusões

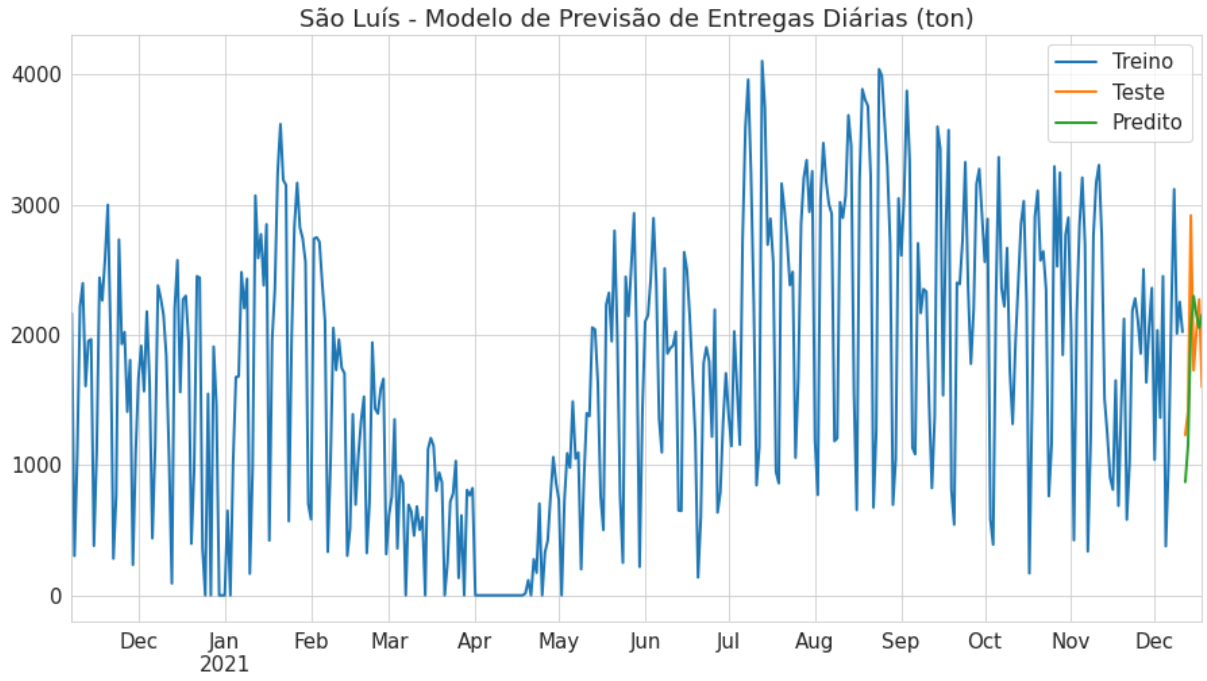

Figura 6 – Bases de treino e teste para validação do modelo preditivo

Fonte: Elaborado pelo autor

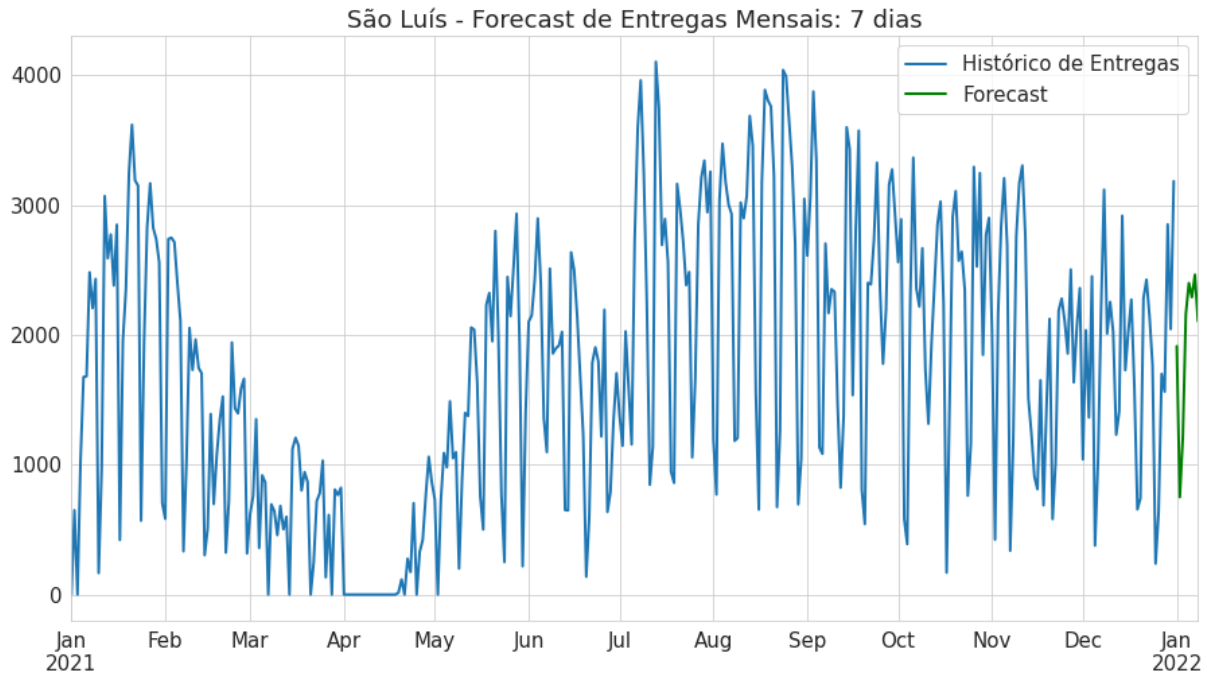

Figura 7 – Forecast para entregas diárias dos próximos 7 dias

Fonte: Elaborado pelo autor

obtidas da Figura 8. Ademais, o teor residual é bastante proeminente, mostrando a relevância do comportamento aleatório.

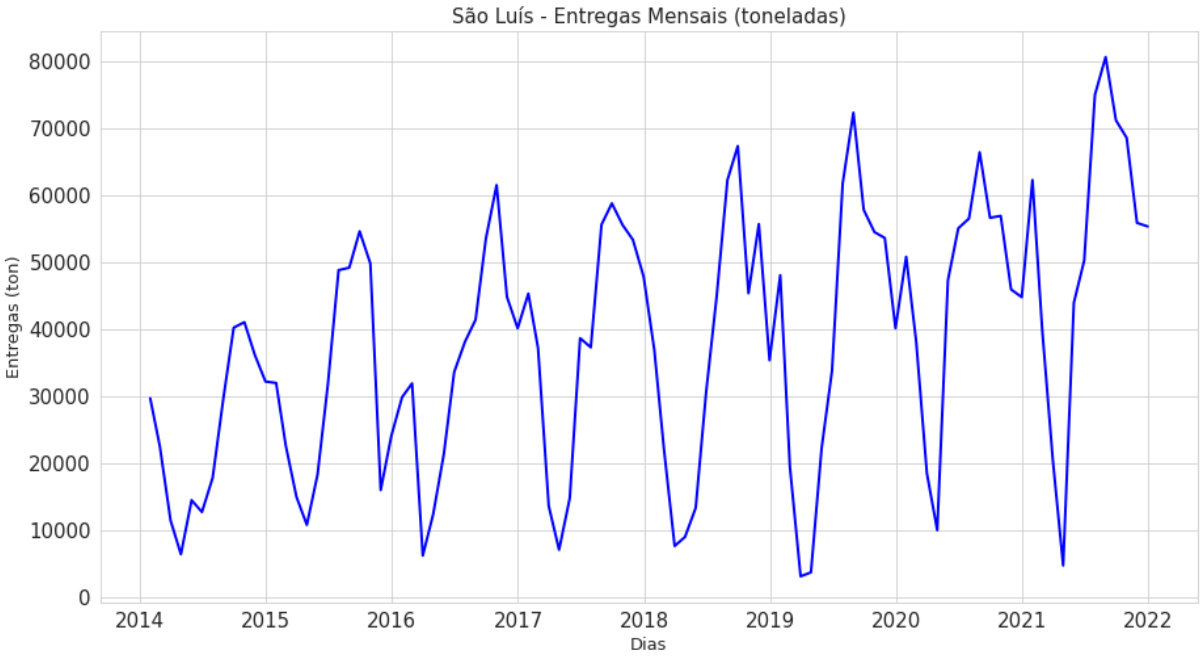

Figura 8 – Entregas mensais em São Luís (toneladas)

Fonte: Elaborado pelo autor

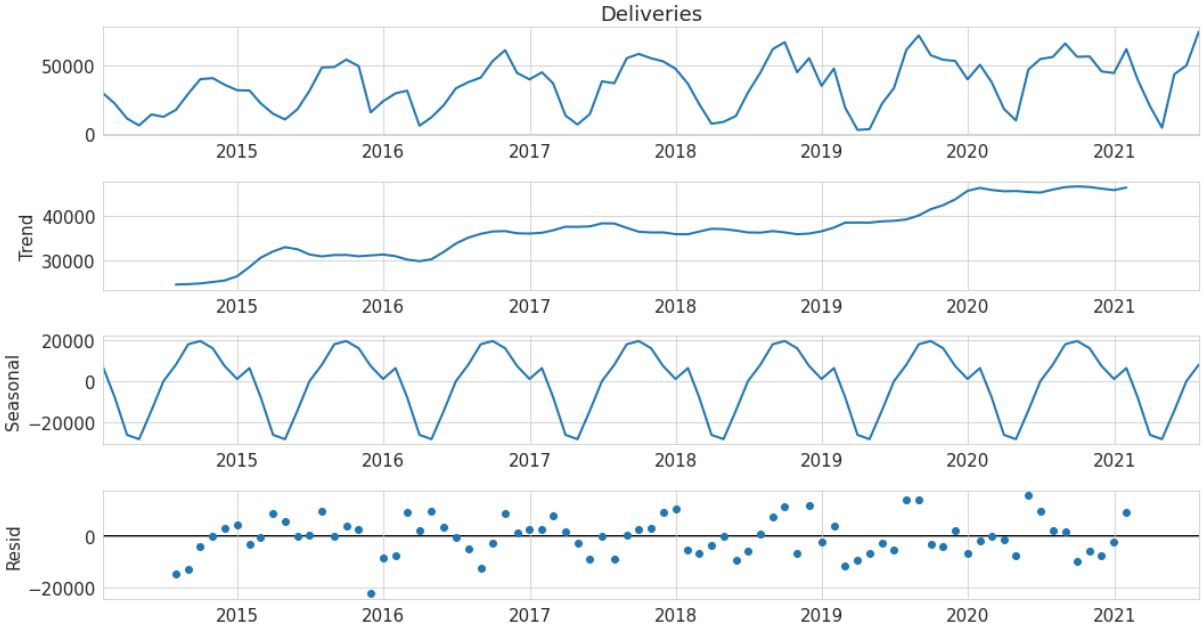

Figura 9 – Decomposição das componentes da série temporal de entregas mensais

Fonte: Elaborado pelo autor

Os correlogramas (figuras 10 e 11) indicam a existência de correlação entre a variável aleatória e sua defasagem, apresentando um comportamento senoide em torno de $lag = 6$, o que mostra justamente a inversão do comportamento de entregas, isto é, a saída de um período de safra para um período de entressafra. Ademais, a estatística de

Dickey-Fuller indica **p-Valor: 0.892696**. Desse modo, não é possível aceitar a hipótese de estacionariedade da série temporal, o que indica que um possível modelo de previsão incorpora a componente integrada, realizando a busca pela minimização de AIC através da série das diferenças. Efetivamente, a série das diferenças apresenta **p-Valor: 3.10e-06**, sendo, portanto, estacionária.

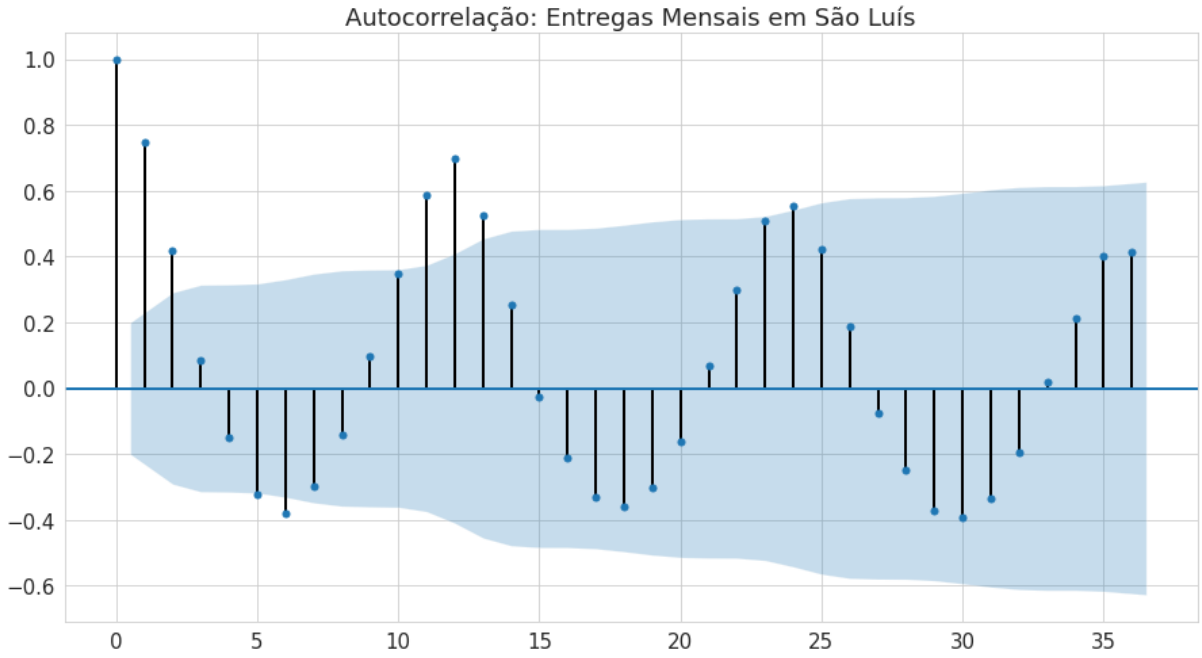

Figura 10 – Autocorrelação de entregas mensais em São Luís

Fonte: Elaborado pelo autor

4.2.3 Modelagem Preditiva e Forecast

O modelo que minimiza o AIC através do *auto_arima* é o **sARIMA(1,0,0)(0,1,1)[12]**. A Figura 12 evidencia o fit que estima os parâmetros da equação 2.28. O erro percentual médio absolute é de MAPE $=13,60\%$. O modelo criado é utilizado em um forecast de 5 meses conforme a Figura 13.

4.3 São Luís: Volume de Entregas Mensais de Matéria-Prima

4.3.1 Visualização de Dados

O volume de entregas de fertilizantes nas indústrias está diretamente relacionado com a estratégia de abastecimento adotado pela companhia, buscando melhor posição comercial no mercado interno. Desse modo, é certo que existe uma correlação entre posição de estoque e consumo de matéria-prima.

Como observado na seção 4.2.2, existe uma tendência de crescimento na série temporal de entregas totais, além de fortes componentes sazonal e aleatória. A figura

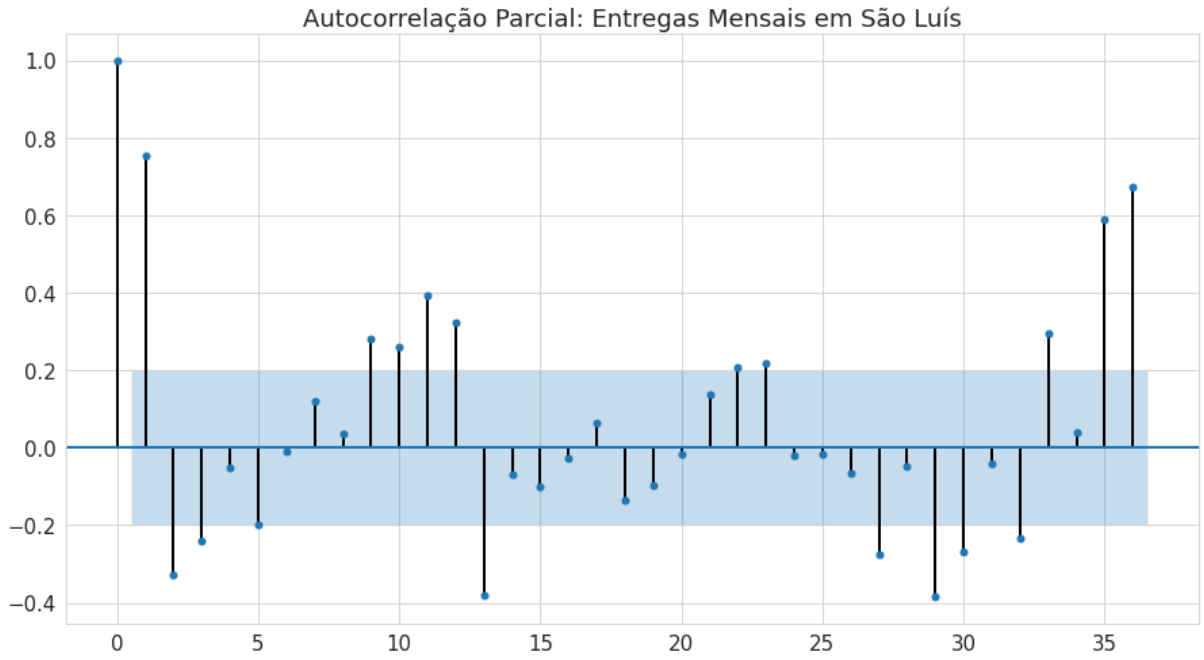

Figura 11 – Autocorrelação parcial de entregas mensais em São Luís

Fonte: Elaborado pelo autor

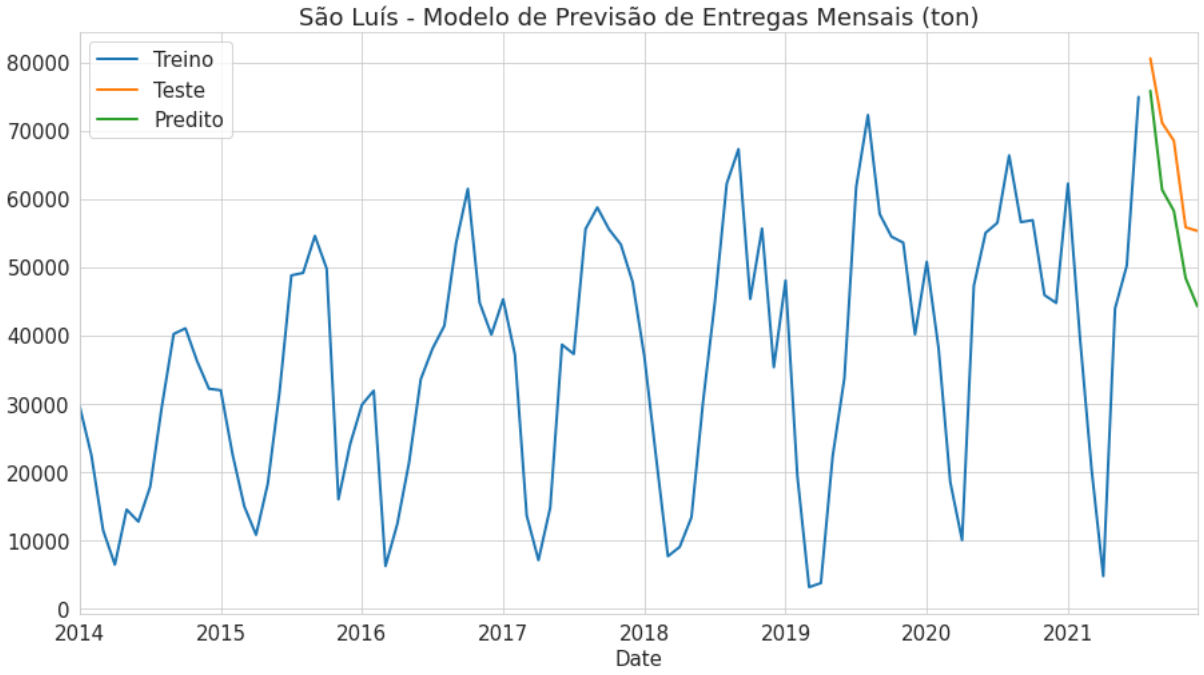

Figura 12 – Bases de treino e teste para validação do modelo preditivo da série temporal de entregas mensais

Fonte: Elaborado pelo autor

14 traz uma comparação entre entregas e consumo. De maneira isolada, comparando-se as entregas com os consumos de Cloreto de Potássio, é possível notar apresentam

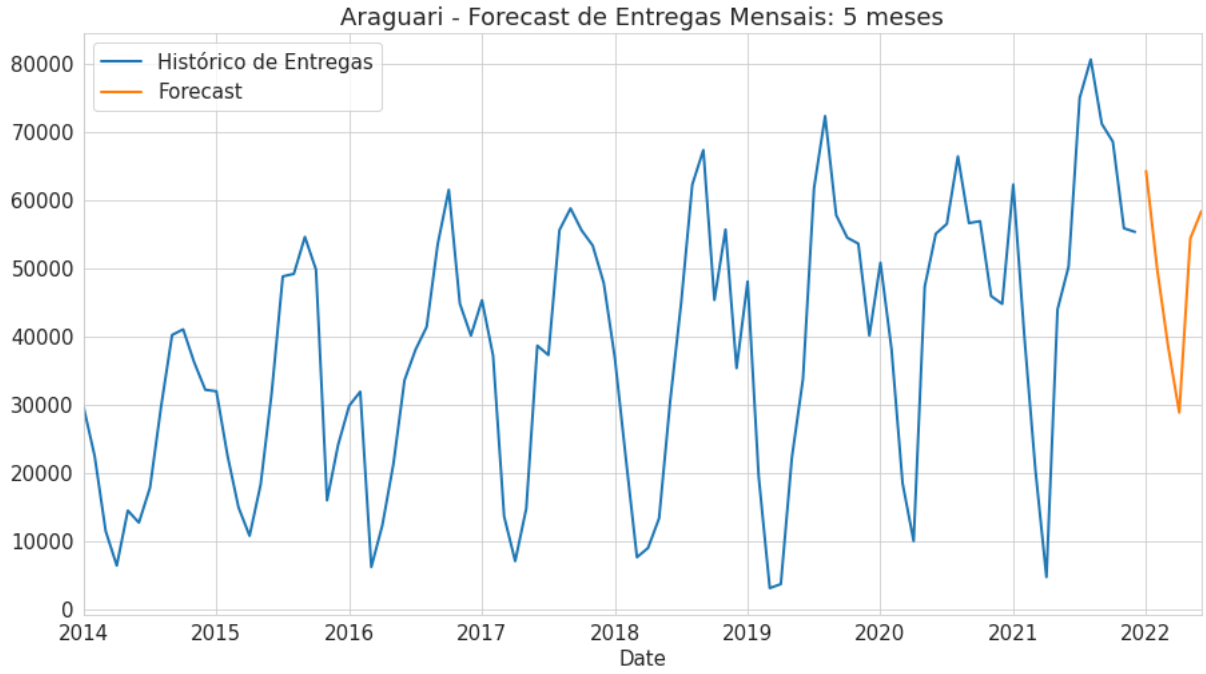

Figura 13 – Bases de treino e teste para validação do modelo preditivo da série temporal de entregas mensais

Fonte: Elaborado pelo autor

um nível elevado ao longo de todo o ano com exceção dos primeiros meses, que são os meses de safrinha e entressafra. Por outro lado, os volumes de Ureia apresentam comportamento oposto, sendo proeminentes durante a safrinha e final da safra. Esse comportamento é esperado e característico do negócio. Por fim, os fertilizantes fosfatados Fosfato Monoamônio (MAP), Super Fosfato Triplo (TSP) e Super Fosfato Simples (SSP) possuem comportamento semelhante ao do Cloreto primariamente por compor formulações que são predominantes durante a safra. Entretanto, ao contrário do Cloreto, possuem intervalo de consumo mais abrangente, englobando ambos os períodos de safra e safrinha.

4.3.2 Análise de Correlação e Modelagem Preditiva: Cloreto

Para os consumos de Cloreto de Potássio, percebe-se com clareza a existência de sazonalidade nos dados como mostra a figura 15. No entanto, não é perceptível a existência de tendência na série temporal. Efetivamente, no ano de 2018 houve um decrescimento proeminente. Esse padrão foi revertido em 2019, como mostrado pela decomposição de série temporal na figura 16.

O teste de Dickey-Fuller indica um **p-Valor: 0.65**, logo, não se tratando de uma série estacionária. No entanto, para a série das primeiras diferenças, **p-Valor: 1.24e-09**, isto é, atinge-se a estacionariedade com a série de diferenças, portanto, corroborando a ideia de que existe uma componente integrada. Ao se utilizar o *auto_arima*, obtemos um

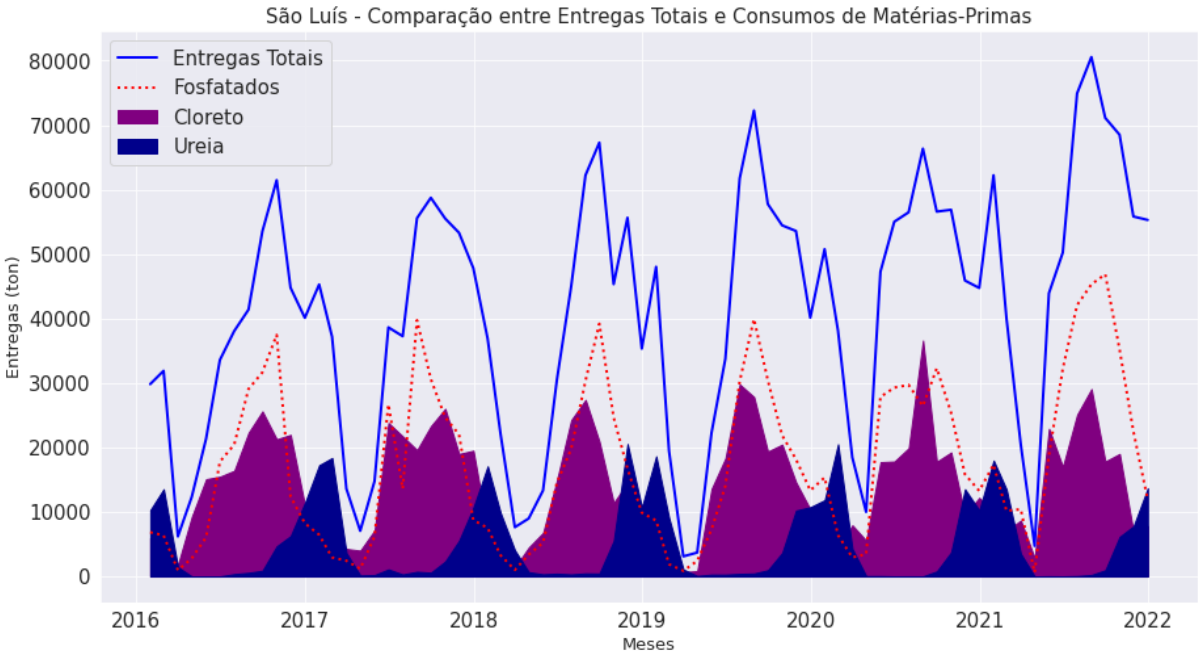

Figura 14 – Comparação entre Entregas Mensais e Consumos de Matérias-Primas (toneladas)

Fonte: Elaborado pelo autor

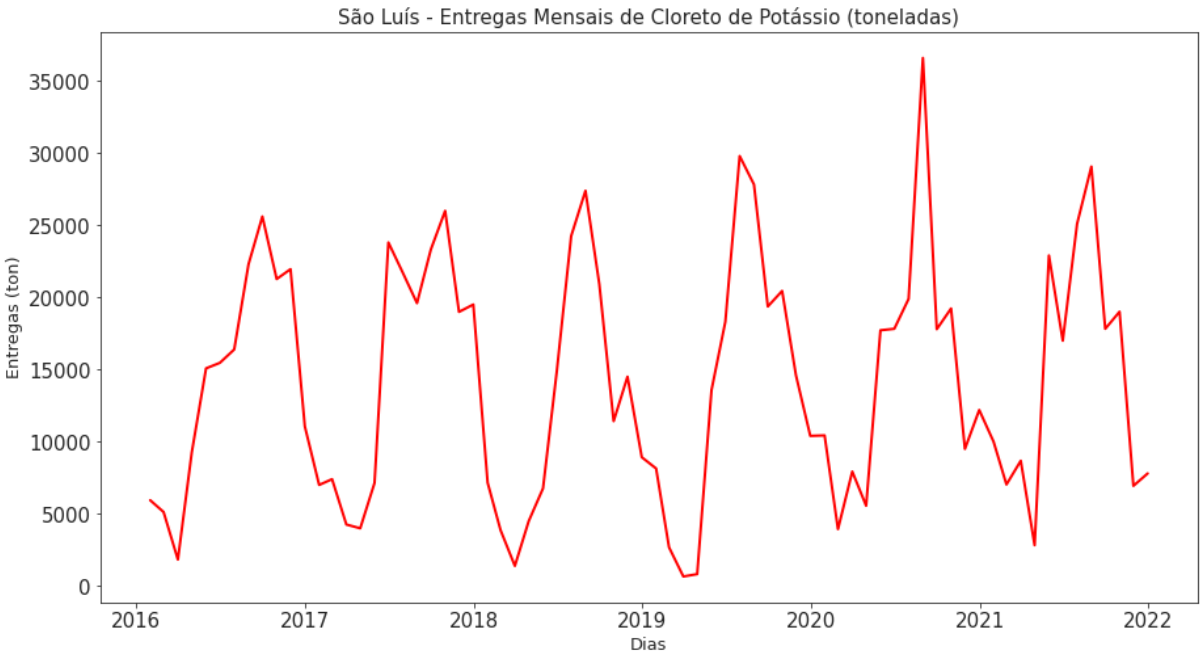

Figura 15 – Entregas Mensais de Cloreto em São Luís (toneladas)

Fonte: Elaborado pelo autor

modelo sugerido **sARIMA(0,0,0)(1,0,0)[12]**, para o qual se avalia o MAPE, obtendo-se MAPE $= 21,90\%$. Por outro lado, introduzindo, pela intuição gerada pelos testes de

Dickey-Fuller, a componente integrada, sugere-se o modelo **sARIMA(0,1,0)(1,1,0)[12]** que possue um MAPE = 12, 78%, apresentando, portanto, melhores resultados preditivos.

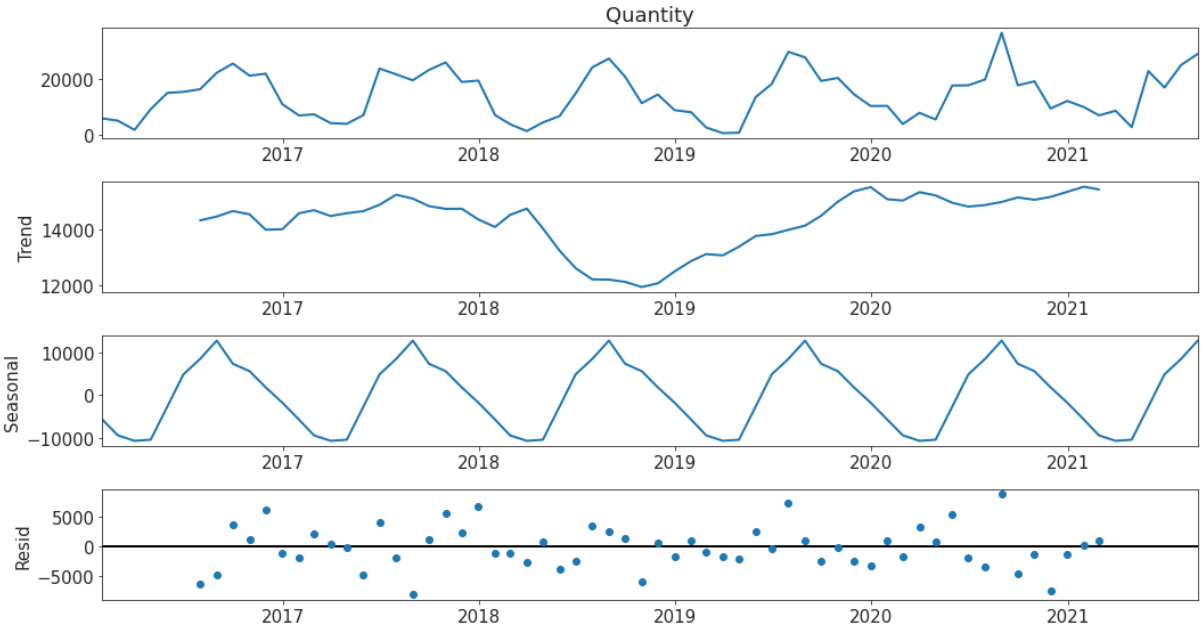

Figura 16 – Decomposição das componentes da série temporal de consumo de Cloreto de Potássio

Fonte: Elaborado pelo autor

O gráfico de autocorrelação (17) da série temporal de consumo de Cloreto evidencia ainda mais claramente a sazonalidade do negócio na medida em que a autocorrelação da variável aleatória é significativa com *lag = 12*. Adicionalmente, o formato senoide revela um pouco do comportamento oscilatório que a série temporal apresenta, porém, também revela autocorrelação negativa em torno de *lag = 6*, indicando que o comportamento de entregas no meio da temporada tem consequências nos valores observados da variável aleatória. Esse fenômeno será abordado em maiores detalhes ao se elucidar a série de consumos de ureia.

A figura 18 mostra a comparação entre os modelos preditivos sARIMA, Holt-Winters e Theta. Uma vez que sARIMA apresentou o menor erro percentual médio absoluto, foi selecionado para predição de valores futuros de consumo de ureia. Efetivamente, graficamente fica evidente que efetivamente um **sARIMA(0,0,0)(1,0,0)[12]** ajusta melhor a série temporal.

4.3.3 Análise de Correlação e Modelagem Preditiva: Ureia

A sazonalidade também é evidente em se tratando da matéria-prima ureia, algo que fica claro já nas primeiras análises da figura 14. Em 19 é possível observar com maior detalhe que os vales no consumo de ureia tem duração mais longa. Além disso, de

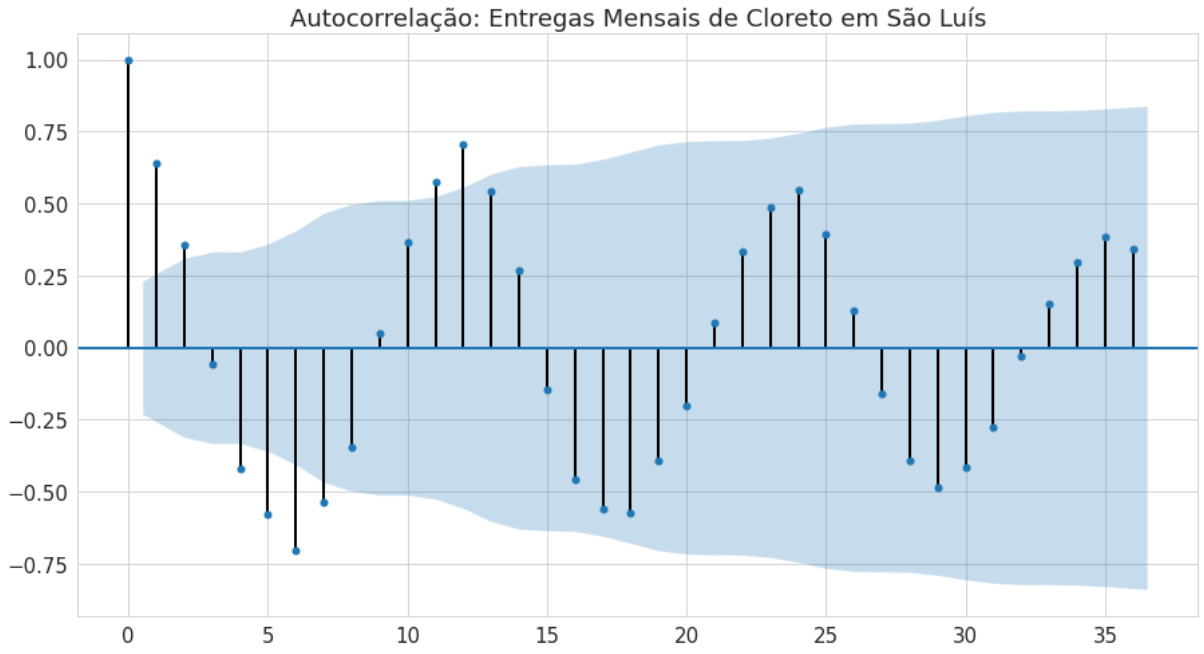

Figura 17 – Autocorrelação para consumo de Cloreto em São Luís

Fonte: Elaborado pelo autor

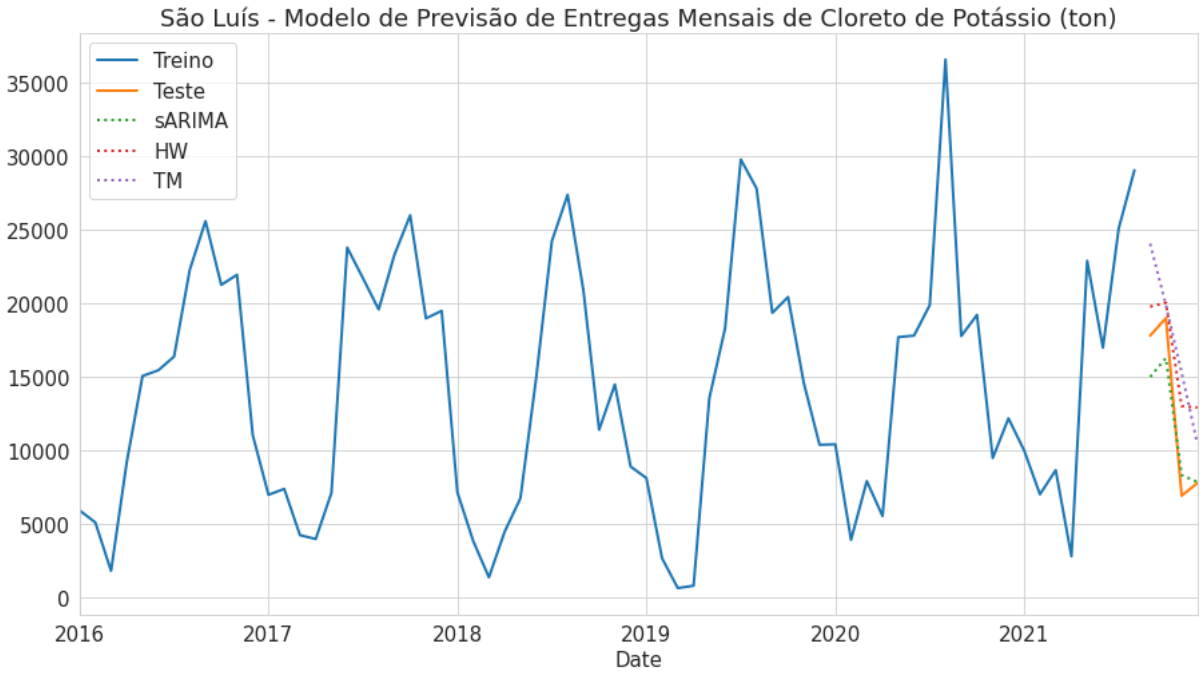

Figura 18 – Comparação entre modelos de previsão de entregas mensais de cloreto

Fonte: Elaborado pelo autor

modo geral, os picos ocorrem 6 meses após o pico de Cloreto. Novamente o gráfico de autocorrelação 20, assim como mostrado na figura 17, apresenta comportamento senoide, revelando sazonalidade e autocorrelação negativa a $lag = 6$. O que se depreende, portanto,

é que o perfil de entregas apresentado pela fábrica de São Luís com uma defasagem de 6 meses tem significância para o valor atual da variável aleatória. Ou seja, em uma situação em que haja antecipação das entregas da safrinha, espera-se naturalmente uma redução nos volumes efetivos durante a período da temporada. Esse situação foi observada durante o final da safra de 2021.

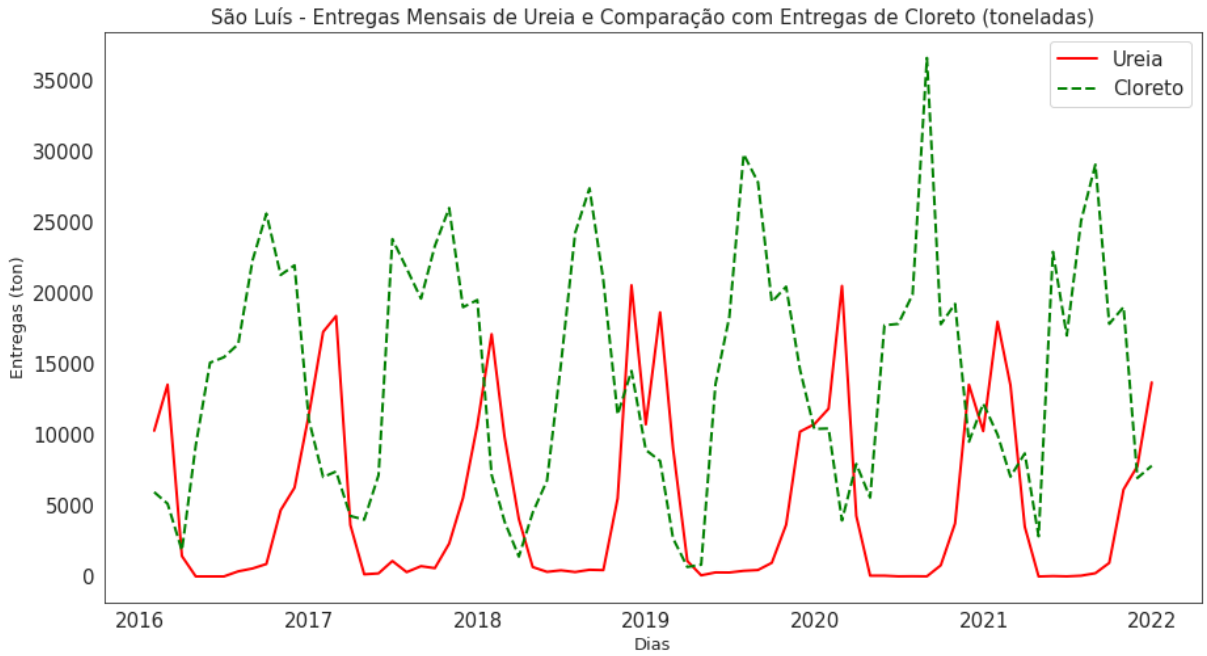

Figura 19 – Comparação entre consumo de ureia e cloreto em São Luís (toneladas)

Fonte: Elaborado pelo autor

Os testes de estacionariedade de Dickey-Fuller evidenciam para a série temporal de ureia **p-Valor: 0.01**, atestando para sua estacionariedade. No entanto, ao aplicar o *auto_arima*, o modelo que minimiza o AIC é **sARIMA(0,0,0)(1,0,2)[12]**. No entanto, esse modelo gera MAPE $= 100,00\%$. Também foi testado Holt-Winters, obtendo um MAPE $= 29,45\%$. Por fim, testa-se um modelo Theta padrão, para o qual MAPE $= 46,74\%$. Portanto, o modelo escolhido para descrever o consumo de Ureia foi um modelo Holt-Winters cujo fit ao modelo pode ser observado na figura 21 e o forecast através da figura 21. Um ponto importante de ser comentado é que, devido à antecipação das entregas de safrinha de 2021, os volumes consumidos de Ureia foram também antecipados, gerando um valor observado maior do que se esperaria. Desse modo, considerando esse ponto atípico, o modelo de Holt-Winters apresenta um MAPE um pouco maior, calculado a MAPE $= 39,18\%$.

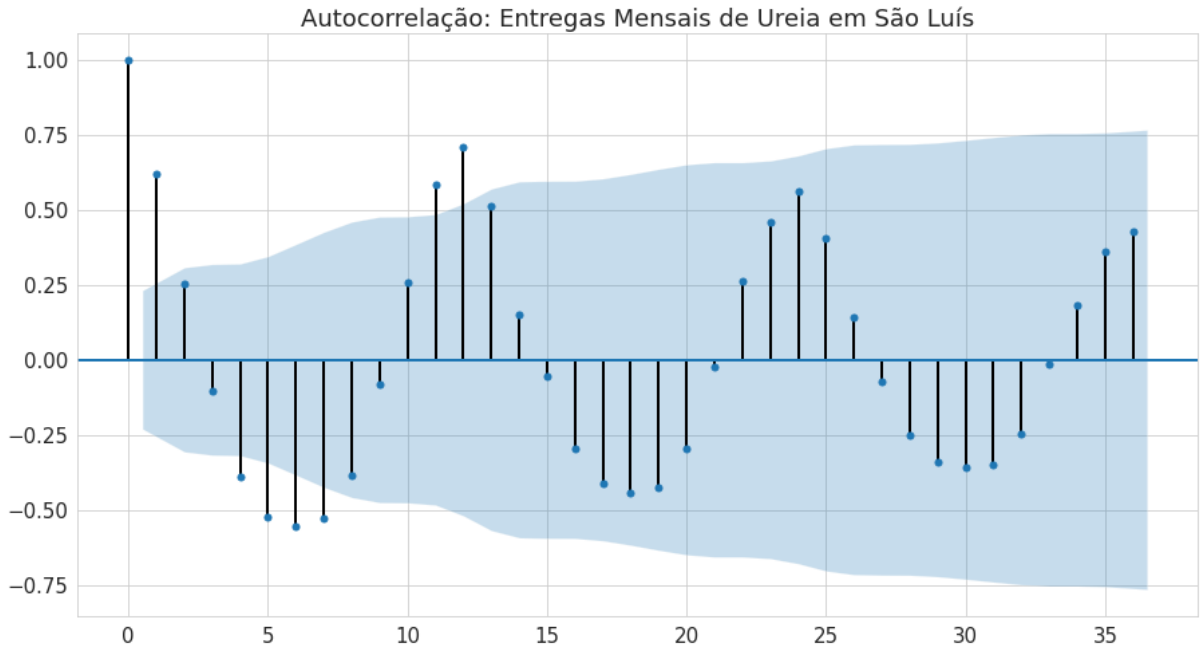

Figura 20 – Autocorrelação para consumo de ureia em São Luís

Fonte: Elaborado pelo autor

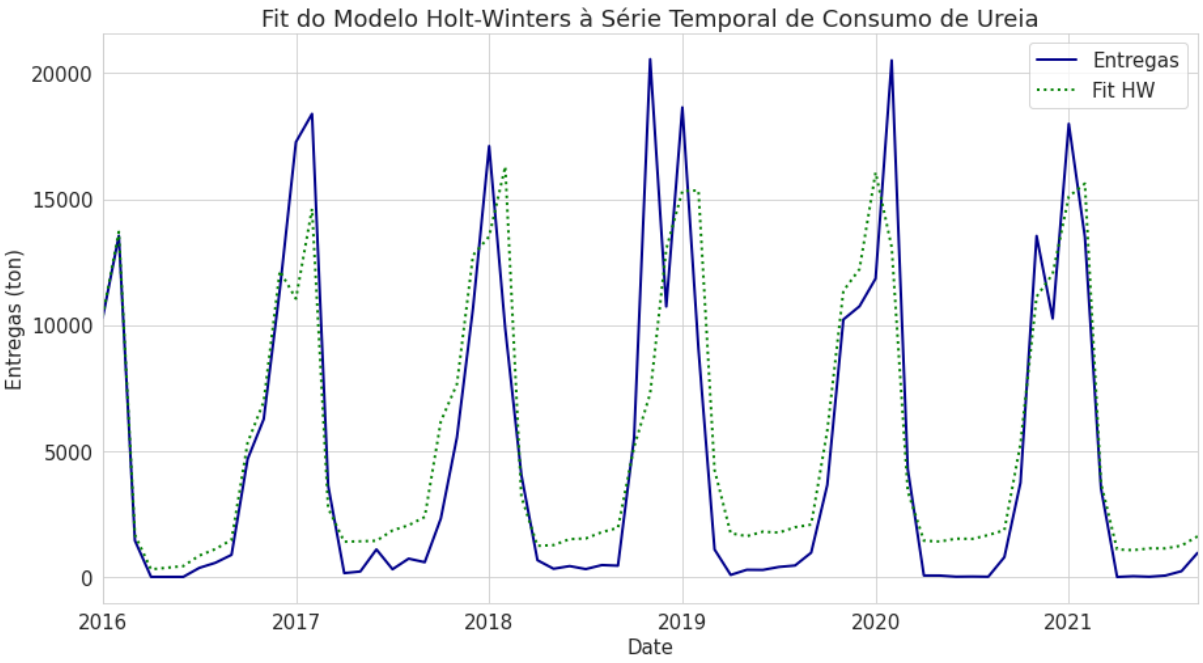

Figura 21 – Fit entre valores observados e modelo Holt-Winters

Fonte: Elaborado pelo autor

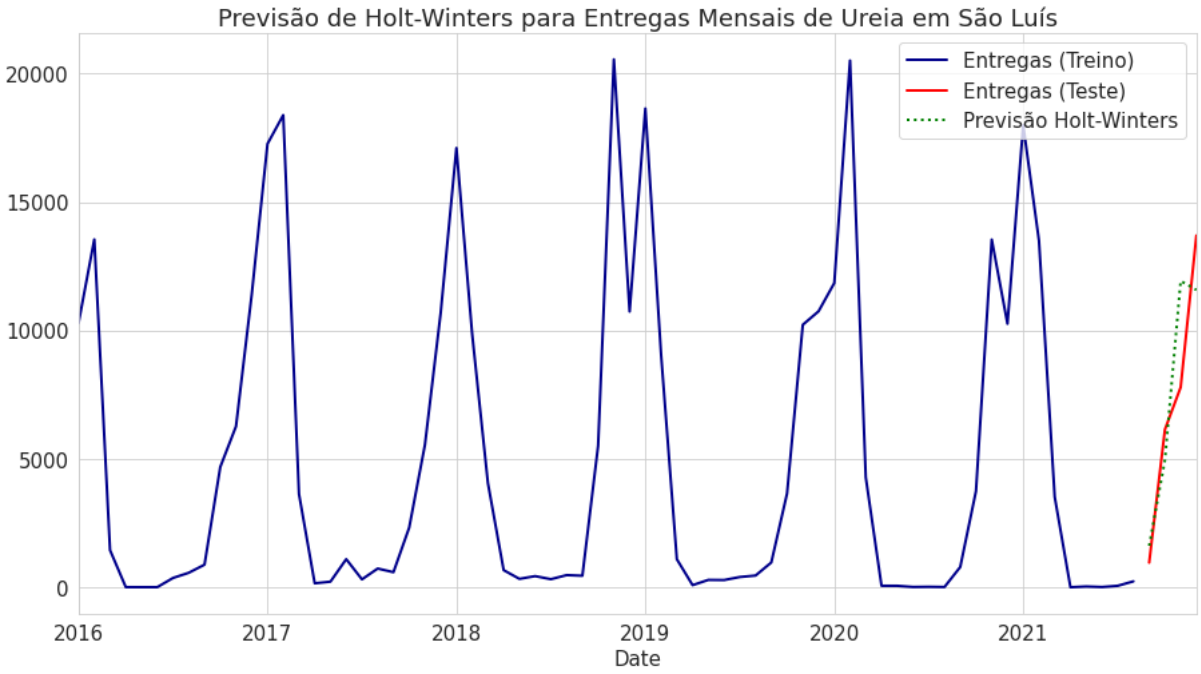

Figura 22 – Previsão de Holt-Winters para entregas mensais de ureia em São Luís

Fonte: Elaborado pelo autor

5 CONCLUSÃO

O estudo e compreensão dos comportamentos apresentados por séries temporais é um tema que evidentemente ganha notoriedade devido à tendência que técnicas computacionais de predição ganham relevância na tomada de decisão das empresas. O entendimento do objeto de observação temporal como uma variável aleatória em um espaço amostral ω que se projeta ao longo do tempo é central para devida elucubração desses fenômenos, permitindo uma formalização matemática que baliza sua interpretação. Dessa forma, é possível utilizar bases históricas para promover estudos preditivos de variáveis de interesse.

A estrutura de tomada de decisão orientada à integração entre os departamentos de uma organização formalmente edificada em torno do processo do S&OP naturalmente envolve a constante atividade de criar um plano de execução para operações em que se tenta maximizar a quantidade de informação coletada de todas as áreas de uma empresa. Entende-se, dessa forma, que a decisão considera a maior quantidade de cenários possível. Entretanto, é inevitável que exista um viés departamental. Desse modo, é extremamente útil para o processo do S&OP que existam ferramentas de forecast auxiliares independente de viéses departamentais.

Em linhas gerais, os modelos de aprendizado dinâmico para previsão de entregas de fertilizantes e consumo de matéria-priima, tais como Cloreto de Potássio e Ureia, são capazes de produzir modelos preditivos robustos o suficiente para utilização em previsão de demanda. Do ponto de vista de negócio, foi possível aplicar técnicas efetivamente nos ciclos do SOP de modo criar uma técnica de forecast para fins comparativos entre inputs de negócio e predições independentes.

Para este trabalho, estudou-se em particular o comportamento histórico de entregas na fábrica de São Luís. Primeiramente, analisou-se os volumes de entregas diárias de fertilizantes. Em seguida, os volumes de entregas mensais. Por fim, os consumos de cloreto e ureia mensais nessa fábrica, de modo a entender os comportamentos sazonais relacionados à operação de produção de fertilizantes.

Em se tratando dos volumes de entrega de fertilizantes, percebe-se como evidenciado pela figura 1 que existe uma sazonalidade anual para a série de entregas, algo que também é corroborado pela figura 8. Isto é, a divisão entre safra, safrinha e entressafra tradicionalmente considerada no mercado de fertilizantes fica evidente em ambas as séries de entregas diárias e mensais. A figura 9 corrobora essa ideia, na medida em que fica evidente a existência clara de componentes tendência, sazonalidade e de médias móveis. Além disso, a nível semanal também existem oscilações sazonais de produção, haja visto que existem turnos produtivos definidos, com paradas semanais para limpeza. A figura 3 realiza

uma decomposição da série de entregas diárias, comprovando as observações da visualização de dados e mostrando que existe a sazonalidade semanal. Os fatores residuais são bastante proeminentes em ambos os casos. No nível diário, a análise de autocorrelação (corroborada pelo correlograma da figura 4), evidêencia alta correlação com elevado nível de significância até $lag = 40$, aproximadamente. O teste de Dickey-Fuller revelou um **p-valor = 0,001123**, aceitando a hipótese de estacionariedade da série. O modelo sARIMA que minimiza o AIC é o **sARIMA(6,1,0)(1,0,2)[7]**, apresentando MAPE = $22,99\%$. Por outro lado, para aa série de entregas mensais o correlograma da figura 9 define bem claramente a existência de tendência de crescimento no volume de entregas, bem como a sazonalidade anual e a existência de fator residuar. Justamente por isso, a estatística de Dickey-Fuller foi aferida em **p-valor = 0.892696**, rejeitando a hipótese de estacionariedade, como esperado pela análise da decomposição da série temporal. No entanto, a série das diferenças apresenta **p-valor = 3.10e-06**, tornando-se estacionária. Conclue-se, portanto, que deve exister uma componente integrada em um modelo sARIMA. Efetivamente, o modelo que minimiza o AIC é o **sARIMA(1,0,0)(0,1,1)[12]**, apresentando um MAPE = $13,60\%$.

Acompanhando o padrão definido pelas entregas mensais, as formulações produzidas em fábrica assumem, majoritariamente, cloreto de potássio, ureia, MAP, TSP e SSP (os três últimos sendo do grupo dos fosfatados) como matérias-primas. A figura 14 demonstra a relevância de tais insumos para as entregas totais dessa indústria, com especial enfoque na fábrica de São Luís. Percebe-se que, para os 3 períodos característicos da temporada (safra, safrinha e entressafra), durante a safra as formulações com cloreto e fosfatados são predominantes, enquanto durante a safrinha predomina a ureia. Para o cloreto, o teste de Dcikey-Fuller indica **p-valor = 0.65**, rejeitando hipótese de estacionariedade, no entanto, **p-valor = 1.24e-09** para a série de diferenças, sendo, desse modo, estacionária. Isso sugere, por conseguinte, que existe componente integrada em modelos sARIMA. O modelo que minimiza o AIC sugerido pelo *auto_arima* é **sARIMA(0,0,0)(1,0,0)[12]** com MAPE = $21,90\%$. No entanto, pela intuição de existência de componente integrada, sugere-se o modelo **sARIMA(0,1,0)(1,1,0)[12]**, que apresentou MAPE = $12,78\%$. O correlograma apresentado pela figura 17 evidencia além da sazonalidade a existência de correlação negativa com $lag = 6$. Isto é reforçado pelo correlograma da figura 20, em que o padrão demonstrado para ureia apresenta o mesmo perfil. Efetivamente, estando no pico da safrinha, para o consumo de ureia, $lag = 6$ significaria o momento oposto - o pico da safra. No entanto, o modelo sARIMA que minimizam o AIC para ureia é o **sARIMA(0,0,0)(1,0,2)[12]**, porém apresentou uma performance ruim ao não captar os longos vales no consumo de ureia. O melhor fit ocorreu utilizando-se Holt-Winters com MAPE = $29,45\%$.

Os modelos de aprendizado dinâmico foram capazes de produzir resultados satisfatórios para a predição dos volumes de entregas diários e principalmente mensais. Além disso, a capacidade de realizar predição de valores de consumo de matéria-prima representa

um salto na qualidade da validação de informações para o ciclo do S&OP. Em especial, o modelo de previsão de entregas mensais está sendo utilizado na avaliação das informações oriendas do ciclo. Além disso, as consultas e relatórios criados para criar os datasets representam uma evolução na qualidade da informação gerada.

Projetos futuros deverão se concentrar em estudar técnicas de aprendizado de máquina para avaliar se os resultados produzidos representam melhoria. Além disso, incluir informações relacionadas a disponibilidade de estoque, situação de carteira comercial e histórico de corredor logístico produziram mais parâmetros para treinar algoritmos de aprendizado de tal modo que seja possível criar um modelo de predição de entregas. Esse modelo deverá ser explorado em relação a si mesmo deprezando o acréscimo de estoque, carteira e logística e em relação aos modelos de aprendizado dinâmico para avaliar se existe ganho preditivo pela adoção dessa técnica. Iniciativas internas da Eurochem Fertilizantes Tocantins visarão buscar estas respostas.

REFERÊNCIAS

ANDRADE, L. **Modelos de inteligência computacional para previsão de demanda desagregada em cadeias varejistas do setor de bens de consumo**. 2020. Dissertação (Mestrado) — Escola Politécnica da Universidade de São Paulo, Bambuí, 2020.

ASSIMAKOPOULOS, V.; NIKOLOPOULOS, K. The theta model: a decomposition approach to forecasting research. **International Journal of Forecasting**, v. 16, n. 4, p. 521–530, 2000.

ATHANASOPOULOS, G.; HYNDMAN, R. J.; WU, D. C. The tourism forecasting competition. **International Journal of Forecasting**, v. 27, n. 1, p. 79–90, 2011.

BOX, G. E. P.; JENKINS, G. M. San Francisco: Holden-Day, 1970.

BOX, G. E. P. et al. **Linear Nonstationary Models**. New Jersey: Wiley Sons, 2016. 88-125 p.

BROCKWELL, P. J.; DAVIS, R. A. **Time Series: Theory and Methods**. New York: Springer, 1991.

BRUNELLE, T. et al. Evaluating the impact of rising fertilizer prices on crop yields. **Agricultural Economics**, v. 46, n. 5, p. 653–666, 2015.

CASELLA, G.; BERGER, R. L. **Statistical Inference**. Pacific Grove: Duxbury, 2001.

EHLERS, R. S. **Análise de Séries Temporais**. São Carlos: Ricardo Sandes Ehlers, 2009.

FIORUCCI, J. A. **Time series forecasting: advances on Theta method**. 2016. Dissertação (Mestrado) — Universidade Federal de São Carlos, São Carlos, 2016.

FIORUCCI, J. A. et al. Models for optimising the theta method and their relationship to state space models. **International Journal of Forecasting**, v. 32, n. 1, p. 1151–1161, 2016.

FISHWICK, P. A. Neural network models in simulation: a comparison with traditional modeling approaches. In: PROCEEDINGS OF THE 1989 WINTER SIMULATION CONFERENCE. Washington, 1989.

HYNDMAN, R. J.; BILLAH, B. Unmasking the theta method. **International Journal of Forecasting**, v. 19, p. 287–290, 2003.

HYNDMAN, R. J.; KOEHLER, A. Another look at measures of forecast accuracy. **International Journal of Forecasting**, v. 22, n. 4, p. 679–688, 2006.

JúNIOR, D. S. de O. S.; OLIVEIRA, J. F. L. de; NETO, P. S. G. de M. An intelligent hybridization of arima with machine leraning models for time series forecasting. **Knowledge-Based Systems**, v. 175, n. 1, p. 72–86, 2019.

KHASHEI, M.; BIJARI, M. A novel hybridization of artificial neural networks and arima models for time series forecasting. **Applied Soft Computing**, v. 11, n. 1, p. 2664–2675, 2011.

KIHORO, J. M.; OTIENO, R. O.; WAFULA, C. Seasonal time series forecasting: a comparative study of arima and ann models. **African Journal of Science and Technology**, v. 5, n. 2, p. 41–49, 2004.

KONING, A. J. et al. The m3 competition: Statistical tests of the results. **International Journal of Forecasting**, v. 21, n. 3, p. 397–409, 2005.

MORETTING, P. A.; TOLOI, C. M. C. **Análise de Séries Temporais**. São Paulo: Blucher, 2004.

OGINO, C. M. et al. Poder de compra, preço e consumo de fertilizantes minerais: uma análise para o centro-oeste brasileiro. **Revista de Economia e Sociologia Rural**, v. 59, n. 1, p. 1–19, 2021.

PEDROSO, C. B.; SILVA, A. L. da; TATE, W. L. Sales and operations planning (s&op): Insights from a multi-case study of brazilian organizations. **International Journal of Production Economics**, v. 182, n. 1, p. 213–229, 2016.

TANG, Z.; ALMEIDA, C.; FISHWICK, P. A. Time series forecasting using neural networks vs. box-jenkins methodology. **Simulation**, v. 57, n. 5, p. 303–310, 1991.

The World Bank. **World Development Indicators**. 2021. Disponível em: <http://databank.worldbank.org/data/>. Acesso em: 23/03/2011.

The World Bank Data Catalog. **Agriculture, Forestry, And Fishing, Value Added (% of GDP)**. 2021. Disponível em: <https://datacatalog.worldbank.org/agriculture-forestry-and-fishing-value-added-gdp-0>. Acesso em: 23/03/2011.

THOMé, A. M. T. et al. Sales and operations planning: A reserach synthesis. **International Journal of Production Economics**, v. 138, n. 1, p. 1–13, 2012.

WALLACE, T. F. **Planejamento de Vendas e Operações: guia prático**. São Paulo: IMAM, 2012. 55-62 p.

WANG, L. et al. An arima-ann hybrid model for time series forecasting. **Systems Research and Behavorial Science**, v. 30, n. 1, p. 244–259, 2013.

YULE, G. U. Why do we sometimes get nonsense-correlations between time series? a study in sampling and the nature of time series. **Journal of Research Statistics Society**, v. 89, n. 1, p. 1–64, 1926.

ZHANG, G. P. Times series forecasting using a hybrid arima and neural network model. **Neurocomputing**, v. 50, n. 1, p. 159–175, 2003.

Apêndices

APÊNDICE A – CÓDIGO SQL EXTRAINDO DADOS HISTÓRICOS DE FATURAMENTO

```
SELECT  ood.organization_code AS "Emp de Faturamento(Code)",
        rctl.warehouse_id AS "Emp de Faturamento(ID)",
        ood.organization_name AS "Emp de Faturamento",
        (SELECT a.registration_number
            FROM xxapps_org_registration_numbers_v a
            WHERE a.organization_id = rctl.warehouse_id
            AND a.registration_code = 'CNPJ') AS "Emp de Faturamento(CNPJ)",
        (SELECT a.registration_number
            FROM xxapps_org_registration_numbers_v a
            WHERE a.Organization_Id = rctl.warehouse_id
            AND a.registration_code = 'IE') AS "Emp de Fat.(Insc. Est.)",
        oola.customer_production_line AS "Emp Produtora(Code)",
        (SELECT organization_id
            FROM org_organization_definitions
            WHERE organization_code = oola.customer_production_line)
            AS "Emp Produtora(ID)",
        (SELECT organization_name
            FROM org_organization_definitions
            WHERE organization_code = oola.customer_production_line)
            AS "Emp Produtora",
        rct.customer_trx_id AS "ID Único da Nota",
        rct.trx_number AS Nota,
        rct.global_attribute3  AS "Serie da Nota",
        jbcte.electronic_inv_access_key AS "Chave de Acesso"
        orf.description AS Transportadora,
        wnd.attribute2 AS Placa,
        null AS "Guia de Trafego",
        hp.party_number AS "Reg de Exerc. do Cli.",
        rctl.line_number AS "Linha da Nota",
        rct.trx_date AS "DT de Emissão Nota",
      , aps.due_date AS "DT de Vencimento"
      , rtt.name AS "Condição de Pagamento"
      , xocged.grp_eco_hdr_id AS "Cod. Grupo Econômico"
      , xocged.name AS "Grupo Econômico"
      , abcr.codigo_interno AS "Cod. Parceiro"
```

```
, hp.party_id
, hp.party_name AS "Razao Cliente"
, abcr.nome_cliente AS Cliente
, abcr.cnpj AS CNPJ
, abcr.endereco||', '||abcr.endereco_1||', '||abcr.bairro AS Endereco
, abcr.cep AS CEP
, abcr.global_attribute3 AS Telefone
, (SELECT customer_class_code
   FROM hz_cust_accounts
   WHERE party_id = hp.PARTY_ID
   AND status = 'A') AS "Tipo de Cliente"
, (SELECT hg.geography_element3_id
   FROM hz_geographies hg
        , hz_geography_identifiers hgi
   WHERE hg.geography_id = hgi.geography_id
   AND hgi.identifier_subtype = 'IBGE'
   AND hgi.identifier_type = 'CODE'
   AND hg.geography_type = 'CITY'
   AND UPPER(hg.geography_name) = UPPER(abcr.cidade)
   AND UPPER(hg.GEOGRAPHY_ELEMENT2) = UPPER(abcr.uf)) AS "Cod. da Cid. Fat."
, abcr.cidade AS "Cidade Faturada"
, (SELECT hg.geography_element2_id
   FROM hz_geographies hg
        , hz_geography_identifiers hgi
   WHERE hg.geography_id = hgi.geography_id
   AND hgi.identifier_subtype = 'IBGE'
   AND hgi.identifier_type = 'CODE'
   AND hg.geography_type = 'STATE'
   AND UPPER(hg.geography_name) = UPPER(abcr.uf)) AS  "Cod. UF Faturada"
, abcr.uf AS "UF Faturada"
, (SELECT DISTINCT ascr.cidade
   FROM apps.xxapps_ar_ship_cust_rel ascr
   WHERE ascr.customer_id = rct.ship_to_customer_id
   AND ascr.site_use_id = rct.ship_to_site_use_id) AS "Cidade de Entrega"
, (SELECT DISTINCT ascr.uf
   FROM apps.xxapps_ar_ship_cust_rel ascr
   WHERE ascr.customer_id = rct.ship_to_customer_id
   AND ascr.site_use_id = rct.ship_to_site_use_id) AS "UF de Entrega"
, (SELECT hgi.identifier_value
```

```
   FROM apps.hz_geographies hg
        , apps.hz_geography_identifiers hgi
   WHERE hg.geography_id = hgi.geography_id
   AND hgi.identifier_subtype = 'IBGE'
   AND hgi.identifier_type = 'CODE'
   AND hg.geography_name =
                  (SElECT DISTINCT ascr.cidade
                  FROM apps.xxapps_ar_ship_cust_rel ascr
                  WHERE ascr.customer_id = rct.ship_to_customer_id
                  AND ascr.site_use_id = rct.ship_to_site_use_id)
   AND hg.object_version_number = 1
   AND hg.geography_element2 =
                  (SELECT DISTINCT ascr.uf
                  FROM apps.xxapps_ar_ship_cust_rel ascr
                  WHERE ascr.customer_id = rct.ship_to_customer_id
                  AND ascr.site_use_id =
                      rct.ship_to_site_use_id)) IBGE,
, EXTRACT(MONTH FROM rct.trx_date) AS "Mes de Emissão"
, EXTRACT(YEAR FROM rct.trx_date) AS "Ano de Emissão"
, TO_CHAR(rct.trx_date, 'MM/YYYY') AS "Período de Emissão"
,  rct.cust_trx_type_id AS "Cod Tp de Op. Fiscal"
,  ctt.name AS "Tp de Op. Fiscal"
, rctl.global_attribute1 AS CFOP
, (SELECT jbap.cfo_description
   FROM jl_br_ap_operations jbap
   WHERE jbap.cfo_code = rctl.global_attribute1) AS
                               "Descrição do CFOP"
, msib.segment1 AS "Cod do Produto"
, msib.description AS Produto
, mic.CATEGORY_CONCAT_SEGS AS NCM
, rctl.UOM_CODE AS UOM
, (SELECT DESCRIPTION
   FROM FND_FLEX_VALUES_VL
   WHERE FLEX_VALUE_SET_ID = 1018812
   AND FLEX_VALUE_MEANING = msib.attribute14) AS Controle
, msib.attribute3 AS "Classificação ANDA"
, rctl.unit_selling_price AS "Valor Unitário"
, rctl.quantity_invoiced AS "QTD Faturada"
, rctl.unit_selling_price * rctl.quantity_invoiced AS "Valor da Nota"
```

```
    , rct.invoice_currency_code AS Moeda
    , CASE
      WHEN rct.invoice_currency_code = 'USD'
           THEN to_number (rctl.attribute12)
           ELSE 1
      END AS "Fator de Conversão Faturamento"
    , TO_DATE(to_char(trunc(rct.trx_date,'MON')), 'DD/MM/YYYY') AS Referência
  , (SELECT NVL(lines.tax_rate, 0)
     FROM apps.ra_customer_trx_lines_all lines
          , apps.zx_lines zl
     WHERE lines.customer_trx_line_id = zl.TRX_LINE_ID
     AND lines.customer_trx_id = zl.trx_id
     AND zl.tax = 'ICMS_C'
     AND lines.customer_trx_line_id = rctl.customer_trx_line_id
     AND lines.customer_trx_id      = rctl.customer_trx_id) AS ICMS
    , (SELECT NVL(lines.tax_rate, 0)
       FROM apps.ra_customer_trx_lines_all lines
            , apps.zx_lines zl
       WHERE lines.customer_trx_line_id = zl.TRX_LINE_ID
       AND lines.customer_trx_id = zl.trx_id
       AND zl.tax = 'PIS_C'
       AND lines.customer_trx_line_id = rctl.customer_trx_line_id
       AND lines.customer_trx_id      = rctl.customer_trx_id) AS PIS
    , (SELECT NVL(lines.tax_rate, 0)
       FROM apps.ra_customer_trx_lines_all lines
            , apps.zx_lines zl
       WHERE lines.customer_trx_line_id = zl.TRX_LINE_ID
       AND lines.customer_trx_id = zl.trx_id
       AND zl.tax = 'COFINS_C'
       AND lines.customer_trx_line_id = rctl.customer_trx_line_id
       AND lines.customer_trx_id      = rctl.customer_trx_id) AS Cofins
    , null AS Frete
    , wnd.fob_code AS "CIF/FOB"
    , ooha.header_id AS "ID OM"
    , xoclp.proposal_number AS "NR Prop de Crédito"
    , xoclp.operation AS "Op de Crédito"
    , ooha.order_number AS "Número do Pedido"
    , NVL(rct.attribute3, rct.attribute4) AS "Versão do Pedido"
    , oola.ORIG_SYS_LINE_REF AS "Referência do Pedido"
```

```
    , oola.line_number AS "Num Linha do Pedido"
    , rctl.reason_code AS "Motivo da Devolução"
    , ooha.ordered_date AS "DT do Pedido"
    , TRUNC(oola.schedule_ship_date) AS "DT de Previsao"
    , oola.ordered_quantity * oola.unit_selling_price AS "Val Total do Pedido"
    , rct.invoice_currency_code AS "Val Moeda do Pedido"
    , oola.attribute12 AS "Ptax Pedido"
    , ctt.name AS "Tipo Titulo"
    , oola.attribute3 AS "Código do Vededor"
    , (SELECT rsa.name
       FROM apps.ra_salesreps_all rsa
       WHERE rsa.salesrep_id =
             replace(replace(replace(oola.attribute3,'/'),':'), ' ')
      ) AS Vendedor
    , oola.attribute5 AS Supervisor
    , oola.attribute7 AS Gerente
    , (SELECT rsa.name
       FROM apps.ra_salesreps_all rsa
       WHERE rsa.salesrep_id  = oola.salesrep_id
       AND rsa.ORG_ID = 82) AS Regional
    , rctl.attribute3 AS "Comissão Vendedor"
FROM apps.ra_customer_trx_all          rct
    , apps.ra_customer_trx_lines_all    rctl
    , apps.org_organization_definitions ood
    , apps.oe_order_headers_all         ooha
    , apps.oe_order_lines_all           oola
    , apps.xxapps_ar_bill_cust_rel      abcr
    , apps.mtl_system_items_b           msib
    , apps.ra_cust_trx_types_all        ctt
    , Hz_parties                        hp
    , ra_terms_tl                       rtt
    , jl_br_customer_trx_exts           jbcte
    , xxapps_om_ctrl_lim_proposal       xoclp
    , xxapps_om_crll_gr_ec_lin          xocgel
    , xxapps_om_crll_gr_ec_hdr          xocged
    , apps.mtl_parameters               mp
    , apps.ar_payment_schedules_all     aps
    , apps.mtl_item_categories_v        mic
    , apps.wsh_new_deliveries_v         wnd
```

```
    , apps.wsh_carriers                wc
    , apps.org_freight                 orf
WHERE rct.customer_trx_id           = rctl.customer_trx_id
  AND rct.org_id                    = rctl.org_id
  AND rct.status_trx                = 'OP'
  AND rctl.warehouse_id             = ood.organization_id
  AND rct.sold_to_customer_id       = ooha.sold_to_org_id
  AND rct.interface_header_attribute1 = ooha.order_number
  AND rctl.inventory_item_id        = oola.inventory_item_id
  AND ooha.header_id                = oola.header_id
  AND rctl.interface_line_attribute6 = oola.line_id
  AND rct.sold_to_customer_id       = abcr.customer_id
  AND rct.BILL_TO_SITE_USE_ID       = abcr.site_use_id
  AND rctl.inventory_item_id        = msib.inventory_item_id
  AND rctl.warehouse_id             = msib.ORGANIZATION_ID
  AND rctl.line_type                = 'LINE'
  AND rct.cust_trx_type_id          = ctt.cust_trx_type_id
  AND ctt.type                      = 'INV'
  AND ctt.post_to_gl                = 'Y'
  AND ctt.org_id                    = rct.org_id
  AND hp.party_id                   = abcr.party_id
  AND rct.term_id                   = rtt.term_id (+)
  AND rtt.language                  = 'PTB'
  AND rct.customer_trx_id           = jbcte.CUSTOMER_TRX_ID(+)
  AND ooha.attribute1               = xoclp.proposal_id(+)
  AND abcr.customer_id              = xocgel.customer_id(+)
  AND xocgel.grp_eco_hdr_id         = xocged.grp_eco_hdr_id(+)
  AND NVL(ood.organization_name,'YYY')    NOT LIKE '%DISCRETA%'
  AND mp.organization_id            = rctl.warehouse_id
  AND aps.customer_trx_id (+)       = rctl.customer_trx_id
  AND mic.category_set_name         =  'FISCAL_CLASSIFICATION'
  AND mic.inventory_item_id         =  rctl.inventory_item_id
  AND mic.organization_id           =  rctl.warehouse_id
  AND wnd.delivery_id               =  rct.interface_header_attribute3 (+)
  AND wc.carrier_id                 =  wnd.carrier_id (+)
  AND orf.organization_id           =  rctl.warehouse_id (+)
  AND orf.freight_code              =  wc.freight_code
```

APÊNDICE B – CÓDIGO SQL EXTRAINDO DADOS HISTÓRICOS DE CONSUMO DE MATÉRIA-PRIMA

```
    SELECT DISTINCT wdv.lot_number ,
xop.emp_faturamento_code ,
xop.emp_faturamento_id ,
xop.emp_faturamento ,
xop.emp_faturamento_cnpj ,
xop.emp_fatinsc_est ,
xop.emp_produtora_code ,
xop.emp_produtora_id ,
xop.emp_produtora ,
xop.id_unico_nota ,
xop.trx_line_id ,
xop.nota ,
xop.serie_nota ,
xop.chave_acesso ,
xop.transportadora ,
xop.placa ,
xop.guia_trafego ,
xop.reg_exerc_cli ,
xop.linha_nota ,
xop.dt_emissao_nota ,
xop.hora_emissao ,
xop.dt_vencimento ,
xop.condicao_pagamento ,
xop.cod_grupo_economico ,
xop.grupo_economico ,
xop.cod_parceiro ,
xop.party_id ,
xop.razao_cliente ,
xop.cliente ,
xop.cnpj ,
xop.endereco ,
xop.cep ,
xop.telefone ,
xop.tipo_cliente ,
xop.cod_cid_fat ,
```

```
xop.cidade_faturada ,
xop.cod_uf_faturada ,
xop.uf_faturada ,
xop.cidade_entrega ,
xop.uf_entrega ,
xop.ibge ,
xop.mes_emissao ,
xop.ano_emissao ,
xop.periodo_emissao ,
xop.cod_op_fiscal ,
xop.tp_op_fiscal ,
xop.cfop ,
xop.descricao_cfop ,
xop.cod_produto ,
xop.produto ,
xop.ncm ,
xop.uom ,
xop.controle ,
xop.classificacao_anda ,
xop.valor_unitario ,
xop.qtd_faturada ,
xop.valor_nota ,
xop.moeda ,
xop.fator_conversao_faturamento ,
xop.referencia ,
xop.icms ,
xop.pis ,
xop.cofins ,
xop.frete ,
xop.cif_fob ,
xop.id_om ,
xop.nr_prop_credito ,
xop.op_credito ,
xop.numero_pedido ,
xop.versao_pedido ,
xop.referencia_pedido ,
xop.num_linha_pedido ,
xop.motivo_devolucao ,
xop.dt_pedido ,
```

```
xop.dt_previsao ,
xop.val_total_pedido ,
xop.moeda_funcional ,
xop.ptax_pedido ,
xop.tipo_titulo ,
xop.codigo_vededor ,
xop.vendedor ,
xop.supervisor ,
xop.gerente ,
xop.diretor ,
xop.regional ,
xop.comissao_vendedor ,
op.nro_op ,
op.dt_abertura_op ,
op.dt_conclusao_op ,
Nvl(op.cod_materia_prima, xop.cod_produto) cod_materia_prima ,
Nvl (op.desc_materia_prima, xop.produto) desc_materia_prima ,
op.uso_ingrediente ,
DECODE(OP.NRO_OP, NULL, xop.qtd_faturada,
       NVL(OP.QTD_MATERIA_PRIMA, 1) * xop.qtd_faturada) qtd_materia_prima ,
NVL(op.LOT_NUMBER_MP, wdv.lot_number) LOT_NUMBER_MP ,
op.custo_mp ,
(XXAPPS.XXAPPS_OPM_RETORNA_FORNEC_FABR(1,Nvl(op.LOT_NUMBER_MP, wdv.lot_number),wdv.
op.FORNECEDOR ,
Nvl(gcc2.unit_cost, op.custo_mp) unit_cost

FROM xxapps.xxapps_opm_pre_450 xop ,
apps.wsh_deliverables_v wdv ,
(SELECT glc.organization_id ,
glc.inventory_item_id ,
glc.lot_number ,
glc.onhand_qty ,
glc.unit_cost ,
glc.cost_date
FROM apps.gmf_lot_costs glc
WHERE glc.unit_cost != 0
AND glc.header_id =
(SELECT MAX(b.header_id)
FROM apps.gmf_lot_costs b
```

```
WHERE b.unit_cost > 0
AND b.inventory_item_id = glc.inventory_item_id
AND b.organization_id = glc.organization_id
AND b.lot_number = glc.lot_number
AND b.cost_date <=
(SELECT end_date
FROM apps.gmf_period_statuses
WHERE period_code = '11.2021'
AND calendar_code = '2021'
)
)
) gcc2 ,
(SELECT rct.customer_trx_id,
rctl.interface_line_attribute3,
rctl.interface_line_attribute6,
MAX(wdv.lot_number) lot_number
FROM ar.ra_customer_trx_all rct,
ar.ra_customer_trx_lines_all rctl,
apps.wsh_deliverables_v wdv
WHERE 1 = 1
AND rct.customer_trx_id = rctl.customer_trx_id
AND rct.org_id = rctl.org_id
AND rct.status_trx = 'OP'
AND rctl.line_type = 'LINE'
AND rctl.interface_line_attribute3 = wdv.delivery_id
AND rctl.inventory_item_id = wdv.inventory_item_id
AND rctl.interface_line_attribute6 = wdv.source_line_id
GROUP BY rct.customer_trx_id,
rctl.interface_line_attribute3,
rctl.interface_line_attribute6
) rct_aux ,
(SELECT mtln.LOT_NUMBER ,
mmt.inventory_item_id ,
mmt.ORGANIZATION_ID ,
gbh.batch_no NRO_OP ,
mmt.transaction_id ID_TRANSACAO ,
gbh.actual_start_date DT_ABERTURA_OP ,
gbh.ACTUAL_CMPLT_DATE DT_CONCLUSAO_OP ,
msi1.segment1 COD_MATERIA_PRIMA ,
```

```
SUBSTR(msi1.description,1, 25) DESC_MATERIA_PRIMA ,
msi1.attribute2 USO_INGREDIENTE ,
DECODE(ABS(SUM(mtln1.PRIMARY_QUANTITY)),
       0,1,ABS(SUM(mtln1.PRIMARY_QUANTITY)))
       /DECODE(MAX(gmd.ACTUAL_QTY),0,1,MAX(gmd.ACTUAL_QTY))
AS QTD_MATERIA_PRIMA ,
mtln1.LOT_NUMBER LOT_NUMBER_MP ,
MAX (XXAPPS.XXAPPS_OPM_RETORNA_FORNEC_FABR(1,mtln1.LOT_NUMBER,
     mtln1.inventory_item_id,mtln1.organization_id)) FORNECEDOR ,
(gcc1.unit_cost) custo_mp
FROM gme.gme_batch_header gbh ,
gme.gme_material_details gmd ,
inv.mtl_transaction_lot_numbers mtln ,
inv.mtl_transaction_lot_numbers mtln1 ,
inv.mtl_material_transactions mmt ,
inv.mtl_material_transactions mmt1 ,
apps.mtl_system_items msi ,
apps.mtl_system_items msi1 ,
(SELECT glc.organization_id ,
glc.inventory_item_id ,
glc.lot_number ,
glc.onhand_qty ,
glc.unit_cost ,
glc.cost_date
FROM apps.gmf_lot_costs glc
WHERE glc.unit_cost != 0
AND glc.header_id =
(SELECT MAX(b.header_id)
FROM apps.gmf_lot_costs b
WHERE b.unit_cost > 0
AND b.inventory_item_id = glc.inventory_item_id
AND b.organization_id = glc.organization_id
AND b.lot_number = glc.lot_number
AND b.cost_date <=
(SELECT end_date
FROM apps.gmf_period_statuses
WHERE period_code = '11.2021'
AND calendar_code = '2021'
)
```

```
)
) gcc1
WHERE 1 = 1
AND gbh.batch_id = gmd.batch_id
AND gmd.line_type = 1
AND mmt.TRANSACTION_TYPE_ID IN (44 , 17)
AND gbh.batch_id = mmt.transaction_source_id
AND gmd.ACTUAL_QTY > 0
AND gbh.BATCH_TYPE = 0
AND gmd.inventory_item_id = mmt.inventory_item_id
AND mtln.transaction_id = mmt.transaction_id
AND mtln1.transaction_id = mmt1.transaction_id
AND mmt.inventory_item_id = msi.inventory_item_id
AND mmt1.inventory_item_id = msi1.inventory_item_id
AND mmt.organization_id = msi.organization_id
AND mmt1.organization_id = msi1.organization_id
AND mmt1.TRANSACTION_TYPE_ID IN (43 , 35)
AND mmt.transaction_source_id = mmt1.transaction_source_id
AND gmd.batch_id = mmt1.transaction_source_id
AND gcc1.organization_id (+) = mtln1.organization_id
AND gcc1.inventory_item_id (+) = mtln1.inventory_item_id
AND gcc1.lot_number (+) = mtln1.LOT_NUMBER
GROUP BY mtln.LOT_NUMBER ,
mmt.inventory_item_id ,
mmt.ORGANIZATION_ID ,
gbh.batch_no ,
mmt.transaction_id ,
gbh.actual_start_date ,
gbh.ACTUAL_CMPLT_DATE ,
msi1.segment1 ,
SUBSTR(msi1.description,1, 25) ,
msi1.attribute2 ,
mtln1.LOT_NUMBER ,
(gcc1.unit_cost)
) op ,
(SELECT mtln.LOT_NUMBER ,
mmt.transaction_id ID_TRANSACAO ,
mmt.TRX_SOURCE_LINE_ID
FROM inv.mtl_transaction_lot_numbers mtln ,
```

```
inv.mtl_material_transactions mmt ,
apps.mtl_system_items msi
WHERE mtln.transaction_id = mmt.transaction_id
AND mmt.inventory_item_id = msi.inventory_item_id
AND mmt.organization_id = msi.organization_id
AND NOT EXISTS
(SELECT 1
FROM gme.gme_batch_header gbh
WHERE gbh.batch_id = mmt.transaction_source_id
AND mmt.TRANSACTION_TYPE_ID IN (44 , 17)
)
) SEM_OP
WHERE xop.interface_line_attribute3 = wdv.delivery_id
AND xop.inventory_item_id = wdv.inventory_item_id
AND xop.interface_line_attribute6 = wdv.source_line_id
AND OP.LOT_NUMBER(+) = wdv.lot_number
AND OP.INVENTORY_ITEM_ID(+) = xop.INVENTORY_ITEM_ID
AND OP.ORGANIZATION_ID(+) = case
when xop.EMP_PRODUTORA_ID in ('87', '93', '101')
then xop.warehouse_id
else NVL(xop.Emp_Produtora_Id,xop.warehouse_id) end
AND SEM_OP.TRX_SOURCE_LINE_ID = xop.line_id_om
AND TO_CHAR(xop.interface_line_attribute3) =
    TO_CHAR(rct_aux.interface_line_attribute3)
AND TO_CHAR(xop.interface_line_attribute6) =
    TO_CHAR(rct_aux.interface_line_attribute6)
AND xop.id_unico_nota = rct_aux.customer_trx_id
AND wdv.lot_number = rct_aux.lot_number
AND gcc2.organization_id (+) = NVL(xop.Emp_Produtora_Id,xop.warehouse_id)
AND gcc2.inventory_item_id (+) = xop.inventory_item_id
AND gcc2.lot_number (+) = wdv.lot_number
AND xop.id_unico_nota = 612386 -- in (593989, 634891)
ORDER BY wdv.lot_number,
DESC_MATERIA_PRIMA;
```

APÊNDICE C – PRÉ-TRATAMENTO DOS RELATÓRIOS DE FATURAMENTO

```
etapa1 = Excel.CurrentWorkbook(){[Name="Calendar"]}[Content]
etapa2 = Table.TransformColumnTypes(Fonte,{{"Dia", type date}})
etapa3 = Table.AddColumn(#"Tipo Alterado", "Texto Antes do Delimitador",
         each Text.BeforeDelimiter(Text.From([Dia], "pt-BR"), "/"), type text)
etapa4 = Table.AddColumn(#"Texto Inserido Antes do Delimitador",
         "Texto Entre os Delimitadores",
         each Text.BetweenDelimiters(Text.From([Dia], "pt-BR"),
         "/", "/"), type text)
etapa5 = Table.AddColumn(#"Texto Inserido Entre os Delimitadores",
         "Últimos caracteres", each Text.End(Text.From([Dia],
         "pt-BR"), 4), type text)
etapa6 = Table.ReorderColumns(#"Últimos caracteres inseridos",
         {"Dia", "Últimos caracteres",
         "Texto Entre os Delimitadores", "Texto Antes do Delimitador"})
etapa7 = Table.RenameColumns(#"Colunas Reordenadas",
         {{"Últimos caracteres", "Year"}, {"Texto Entre os Delimitadores",
         "Month"}, {"Texto Antes do Delimitador", "Day"}})
etapa8 = Table.AddColumn(#"Colunas Renomeadas",
         "Date", each Text.Combine({[Year], [Month], [Day]}, "-"), type text)
etapa9 = Table.ReorderColumns(#"Coluna Mesclada Inserida",
         {"Dia", "Date", "Year", "Month", "Day"})
etapa10 = Table.RemoveColumns(#"Colunas Reordenadas1",
          {"Year", "Month", "Day"})
etapa11 = Table.ReorderColumns(#"Colunas Removidas",{"Date", "Dia"})
etapa12 = Table.NestedJoin(#"Colunas Reordenadas2",
          {"Dia"}, fDeliveries, {"Dt. Neg."},
          "fDeliveries", JoinKind.FullOuter)
etapa13 = Table.ExpandTableColumn(#"Consultas Mescladas", "fDeliveries",
          {"Dt. Neg.", "Empresa Faturamento", "Empresa Produtora",
          "Regional", "Qtd. (ton)"}, {"Dt. Neg.", "Empresa Faturamento",
          "Empresa Produtora", "Regional", "Qtd. (ton)"})
etapa14 = Table.Sort(#"fDeliveries Expandido",{{"Dia", Order.Ascending}})
etapa15 = Table.RenameColumns(#"Linhas Classificadas",{{"Dia", "Day"}})
etapa16 = Table.RemoveColumns(#"Colunas Renomeadas1",{"Dt. Neg."})
etapa17 = Table.RenameColumns(#"Colunas Removidas1",
```

```
          {{"Empresa Faturamento", "Invoicing Unit"},
          {"Empresa Produtora", "Production Unit"},
          {"Qtd. (ton)", "Deliveries"}})
etapa18 = Table.DuplicateColumn(#"Colunas Renomeadas2",
          "Deliveries", "Deliveries - Copiar")
etapa19 = Table.RenameColumns(#"Coluna Duplicada",
          {{"Deliveries - Copiar", "Dotted Deliveries"}})
etapa20 = Table.TransformColumnTypes(#"Colunas Renomeadas3",
          {{"Dotted Deliveries", type text}})
etapa21 = Table.ReplaceValue(#"Tipo Alterado1",",",".",
          Replacer.ReplaceText,{"Dotted Deliveries"})
etapa22 = Table.AddColumn(#"Valor Substituído",
          "Date,bUnit,pUnit,Regional,Deliveries", each Text.Combine({[Date],
          [Invoicing Unit], [Production Unit], [Regional],
          [Dotted Deliveries]}, ","), type text)
etapa23 = Table.RenameColumns(#"Coluna Mesclada Inserida1",
          {{"Invoicing Unit", "bUnit"},
          {"Production Unit", "pUnit"}})
etapa24 = Table.AddColumn(#"Colunas Renomeadas4",
          "Filter", each if([Day]<Date.From(DateTime.LocalNow()))
                         then "Sim" else "Não")
etapa25 = Table.SelectRows(#"Personalização Adicionada",
          each ([Filter] = "Sim"))
etapa26 = Table.RemoveColumns(#"Linhas Filtradas",
          {"Filter", "Date", "Day", "bUnit", "pUnit", "Regional",
          "Deliveries", "Dotted Deliveries"})
```

APÊNDICE D – PRÉ-TRATAMENTO DOS RELATÓRIOS DE PRODUÇÃO

```
etapa1 = Excel.CurrentWorkbook(){[Name="Calendar"]}[Content]
etapa2 = Table.TransformColumnTypes(Fonte,{{"Dia", type date}})
etapa3 = Table.NestedJoin(#"Tipo Alterado", {"Dia"}, Production,
         {"Data Faturamento"}, "Production", JoinKind.LeftOuter)
etapa4 = Table.ExpandTableColumn(#"Consultas Mescladas","Production",
         {"Cód. MP", "Matéria-Prima", "Descrição Geral", "Qtd. (ton)",
         "Cód. Empresa Produtora", "Cód. Empresa Faturamento"},
         {"Production.Cód. MP", "Production.Matéria-Prima",
         "Production.Descrição Geral",
         "Production.Qtd. (ton)",
         "Production.Cód. Empresa Produtora",
         "Production.Cód. Empresa Faturamento"})
etapa5 = Table.Sort(#"Production Expandido",{{"Dia", Order.Ascending}})
etapa6 = Table.AddColumn(#"Linhas Classificadas",
         "Texto Antes do Delimitador",
         each Text.BeforeDelimiter(Text.From([Dia],
         "pt-BR"), "/"), type text)
etapa7 = Table.AddColumn(#"Texto Inserido Antes do Delimitador",
         "Texto Entre os Delimitadores",
         each Text.BetweenDelimiters(Text.From([Dia], "pt-BR"),
         "/", "/"), type text)
etapa8 = Table.RenameColumns(#"Texto Inserido Entre os Delimitadores",
         {{"Texto Antes do Delimitador", "Dia.1"},
         {"Texto Entre os Delimitadores", "Mês"}})
etapa9 = Table.AddColumn(#"Colunas Renomeadas1",
         "Ano", each Date.Year([Dia]), Int64.Type)
etapa10 = Table.CombineColumns(Table.TransformColumnTypes(#"Ano Inserido",
          {{"Ano", type text}, {"Mês", type text}, {"Dia.1", type text}},
          "pt-BR"),{"Ano", "Mês", "Dia.1"},
          Combiner.CombineTextByDelimiter("-",
          QuoteStyle.None),"Date")
etapa11 = Table.RemoveColumns(#"Colunas Mescladas",{"Dia"})
etapa12 = Table.RenameColumns(#"Colunas Removidas",{{"Production.Cód. MP",
          "ProductCode"}, {"Production.Matéria-Prima", "ProductDesc"},
          {"Production.Descrição Geral", "ProductGenDesc"},
          {"Production.Qtd. (ton)", "Quantity"},
```

```
          {"Production.Cód. Empresa Produtora", "pUnit"},
          {"Production.Cód. Empresa Faturamento", "fUnit"}})
etapa13 = Table.ReorderColumns(#"Colunas Renomeadas",{"Date", "pUnit",
          "fUnit", "ProductCode", "ProductDesc", "ProductGenDesc", "Quantity"})
etapa14 = Table.TransformColumnTypes(#"Colunas Reordenadas",
          {{"Quantity", type text}})
etapa15 = Table.ReplaceValue(#"TipoAlterado1",",",".",
          Replacer.ReplaceText,{"Quantity"})
etapa16 = Table.ReplaceValue(#"Valor Substituído",",",".",
          Replacer.ReplaceText,{"ProductDesc"})
etapa17 = Table.ReplaceValue(#"Valor Substituído1",",",".",
          Replacer.ReplaceText,{"ProductGenDesc"})
etapa18 = Table.CombineColumns(#"Valor Substituído2",
          {"Date", "pUnit", "fUnit", "ProductCode", "ProductDesc",
          "ProductGenDesc", "Quantity"},Combiner.CombineTextByDelimiter(",",
          QuoteStyle.None),"Date,pUnit,bUnit,ProductCode,ProductDesc,
          ProductGenDesc,Quantity")
```

APÊNDICE E – NOTEBOOK PARA TRATAMENTO DE BASES: S&OP FORECASTING TOOL

```
!pip install statsmodels==0.12.2
!pip install pmdarima

from google.colab import drive
drive.mount('/content/drive')
# Vinculando endereços de drive para query
path = '/content/drive/My Drive/USP/TCC/Database/'
file1 = 'CSV_deliveries.csv'
file2 = 'CSV_production.csv'

# Importanto as bibliotecas
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
# Importando pacotes necessários
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_absolute_percentage_error
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.forecasting.theta import ThetaModel
from statsmodels.tsa.holtwinters import ExponentialSmoothing
from matplotlib import pyplot
from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.tsa.stattools import adfuller
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from pmdarima import auto_arima
%matplotlib inline
# Ignorar mensagens de erro
import warnings
warnings.filterwarnings('ignore')
# Definindo o tamanho e estilo dos gráficos
plt.rc("figure",figsize=(22,6))
plt.rc("font",size=15)
plt.rc("lines",linewidth=2)
sns.set_style("darkgrid")
```

```
## Importação da base de dados de entregas (CSV_deliveries.csv) do ORACLE
df1 = pd.read_csv(path+file1)
df1['Deliveries'].fillna(0,inplace=True)

## Dataframe global
df1['Date'] = pd.to_datetime(df1['Date'])
dm1 = df1.groupby(pd.Grouper(key='Date', freq='M'))['Deliveries'].sum()
#dm.index = dm.index.strftime('%B')
dm1 = pd.DataFrame(dm1)

## Plotar as entregas mensais
sns.set_style('ticks')
mDeliveries = sns.lineplot(data = dm1, x=dm1.index,
                           y = dm1['Deliveries'], color = 'green')
mDeliveries.set_title('Volume de Entregas Mensais (em toneladas)', fontsize = 15)
mDeliveries.set_ylabel('Entregas (ton)', fontsize = 13)
mDeliveries.set_xlabel('Meses', fontsize = 13)
plt.show()

## Importação da base de dados de entregas (CSV_production.csv) do ORACLE
df2 = pd.read_csv(path+file2)
df2['Quantity'].fillna(0,inplace=True)

## Dataframe global
df2['Date'] = pd.to_datetime(df2['Date'])
dm2 = df2.groupby(pd.Grouper(key='Date', freq='M'))['Quantity'].sum()
#dm2.index = dm2.index.strftime('%B')
dm2 = pd.DataFrame(dm2)

## Plotar as entregas mensais
sns.set_style('ticks')
mProduction = sns.lineplot(data = dm2, x=dm2.index,
                           y = dm2['Quantity'], color = 'blue')
mProduction.set_title('Produção Mensal (em toneladas)', fontsize = 15)
mProduction.set_ylabel('Produção (ton)', fontsize = 13)
mProduction.set_xlabel('Meses', fontsize = 13)
plt.show()
```

```
# CSV_deliveries.csv
# Fazendo tratamento da base de entregas (faturamento) para
criar DataFrames para cada unidade produtora

# Criando dataframes para Porto Nacional

df1_PNO = df1[(df1['pUnit'] == 'PNO')]

dm1_PNO = df1_PNO.copy()
dm1_PNO['Date'] = pd.to_datetime(df1_PNO['Date'])
dm1_PNO = df1_PNO.groupby(pd.Grouper(key='Date', freq='M'))['Deliveries'].sum()
#dm.index = dm.index.strftime('%B')
dm1_PNO = pd.DataFrame(dm1_PNO)

df1_PNO = df1_PNO.groupby(by='Date',as_index=False)['Deliveries'].sum()
mask_PNO = df1[(df1['pUnit'].isnull())].drop(['bUnit','pUnit','Regional'],
                                             axis='columns')
df1_PNO = df1_PNO.append(mask_PNO)
df1_PNO['Date'] = pd.to_datetime(df1_PNO['Date'])
df1_PNO = df1_PNO.set_index('Date').sort_index(axis = 0)

idx = pd.date_range(start = df1_PNO.index.min(),
                    end = df1_PNO.index.max(),
                    freq='D')
df1_PNO = df1_PNO.reindex(idx)

df1_PNO[df1_PNO['Deliveries'] < 0] = 0
df1_PNO = df1_PNO.fillna(0)

# Criando dataframes para São Luís

df1_SLO = df1[(df1['pUnit'] == 'SLO')]

dm1_SLO = df1_SLO.copy()
dm1_SLO['Date'] = pd.to_datetime(df1_SLO['Date'])
dm1_SLO = df1_SLO.groupby(pd.Grouper(key='Date', freq='M'))['Deliveries'].sum()
#dm.index = dm.index.strftime('%B')
dm1_SLO = pd.DataFrame(dm1_SLO)
```

```
df1_SLO = df1_SLO.groupby(by='Date',as_index=False)['Deliveries'].sum()
mask_SLO = df1[(df1['pUnit'].isnull())].drop(['bUnit','pUnit','Regional'],
                                             axis='columns')
df1_SLO = df1_SLO.append(mask_SLO)
df1_SLO['Date'] = pd.to_datetime(df1_SLO['Date'])
df1_SLO = df1_SLO.set_index('Date').sort_index(axis = 0)

idx = pd.date_range(start = df1_SLO.index.min(), end = df1_SLO.index.max(), freq='D')
df1_SLO = df1_SLO.reindex(idx)

df1_SLO[df1_SLO['Deliveries'] < 0] = 0
df1_SLO = df1_SLO.fillna(0)

# Criando dataframes para Querência

df1_QRO = df1[(df1['pUnit'] == 'QRO')]

dm1_QRO = df1_QRO.copy()
dm1_QRO['Date'] = pd.to_datetime(df1_QRO['Date'])
dm1_QRO = df1_QRO.groupby(pd.Grouper(key='Date', freq='M'))['Deliveries'].sum()
#dm.index = dm.index.strftime('%B')
dm1_QRO = pd.DataFrame(dm1_QRO)

df1_QRO = df1_QRO.groupby(by='Date',as_index=False)['Deliveries'].sum()
mask_QRO = df1[(df1['pUnit'].isnull())]
           .drop(['bUnit','pUnit','Regional'],axis='columns')
df1_QRO = df1_QRO.append(mask_QRO)
df1_QRO['Date'] = pd.to_datetime(df1_QRO['Date'])
df1_QRO = df1_QRO.set_index('Date').sort_index(axis = 0)

idx = pd.date_range(start = df1_QRO.index.min(), end = df1_QRO.index.max(), freq='D')
df1_QRO = df1_QRO.reindex(idx)

df1_QRO[df1_QRO['Deliveries'] < 0] = 0
df1_QRO = df1_QRO.fillna(0)

# Criando dataframes para Barcarena

df1_BCO = df1[(df1['pUnit'] == 'BCO')]
```

```
dm1_BCO = df1_BCO.copy()
dm1_BCO['Date'] = pd.to_datetime(df1_BCO['Date'])
dm1_BCO = df1_BCO.groupby(pd.Grouper(key='Date',
                          freq='M'))['Deliveries'].sum()
#dm.index = dm.index.strftime('%B')
dm1_BCO = pd.DataFrame(dm1_BCO)

df1_BCO = df1_BCO.groupby(by='Date',as_index=False)['Deliveries'].sum()
mask_BCO = df1[(df1['pUnit'].isnull())]
           .drop(['bUnit','pUnit','Regional'],axis='columns')
df1_BCO = df1_BCO.append(mask_BCO)
df1_BCO['Date'] = pd.to_datetime(df1_BCO['Date'])
df1_BCO = df1_BCO.set_index('Date').sort_index(axis = 0)

idx = pd.date_range(start = df1_BCO.index.min(),
                    end = df1_BCO.index.max(), freq='D')
df1_BCO = df1_BCO.reindex(idx)

df1_BCO[df1_BCO['Deliveries'] < 0] = 0
df1_BCO = df1_BCO.fillna(0)

# Criando dataframes para Fertimaxi

df1_FTX = df1[(df1['pUnit'] == 'FTX')]

dm1_FTX = df1_FTX.copy()
dm1_FTX['Date'] = pd.to_datetime(df1_FTX['Date'])
dm1_FTX = df1_FTX.groupby(pd.Grouper(key='Date',
                          freq='M'))['Deliveries'].sum()
#dm.index = dm.index.strftime('%B')
dm1_FTX = pd.DataFrame(dm1_FTX)

df1_FTX = df1_FTX.groupby(by='Date',as_index=False)['Deliveries'].sum()
mask_FTX = df1[(df1['pUnit'].isnull())]
           .drop(['bUnit','pUnit','Regional'],axis='columns')
df1_FTX = df1_FTX.append(mask_FTX)
df1_FTX['Date'] = pd.to_datetime(df1_FTX['Date'])
df1_FTX = df1_FTX.set_index('Date').sort_index(axis = 0)
```

```
idx = pd.date_range(start = df1_FTX.index.min(),
                    end = df1_FTX.index.max(), freq='D')
df1_FTX = df1_FTX.reindex(idx)

df1_FTX[df1_FTX['Deliveries'] < 0] = 0
df1_FTX = df1_FTX.fillna(0)

# Criando dataframes para Intermarítima

df1_INT = df1[(df1['pUnit'] == 'INT')]

dm1_INT = df1_INT.copy()
dm1_INT['Date'] = pd.to_datetime(df1_INT['Date'])
dm1_INT = df1_INT.groupby(pd.Grouper(key='Date', freq='M'))['Deliveries'].sum()
#dm.index = dm.index.strftime('%B')
dm1_INT = pd.DataFrame(dm1_INT)

df1_INT = df1_INT.groupby(by='Date',as_index=False)['Deliveries'].sum()
mask_INT = df1[(df1['pUnit'].isnull())]
           .drop(['bUnit','pUnit','Regional'],axis='columns')
df1_INT = df1_INT.append(mask_INT)
df1_INT['Date'] = pd.to_datetime(df1_INT['Date'])
df1_INT = df1_INT.set_index('Date').sort_index(axis = 0)

idx = pd.date_range(start = df1_INT.index.min(),
                    end = df1_INT.index.max(), freq='D')
df1_INT = df1_INT.reindex(idx)

df1_INT[df1_INT['Deliveries'] < 0] = 0
df1_INT = df1_INT.fillna(0)

# Criando dataframes para Sinop

df1_SNO = df1[(df1['pUnit'] == 'SNO')]

dm1_SNO = df1_SNO.copy()
dm1_SNO['Date'] = pd.to_datetime(df1_SNO['Date'])
dm1_SNO = df1_SNO.groupby(pd.Grouper(key='Date', freq='M'))['Deliveries'].sum()
```

```
#dm.index = dm.index.strftime('%B')
dm1_SNO = pd.DataFrame(dm1_SNO)

df1_SNO = df1_SNO.groupby(by='Date',as_index=False)['Deliveries'].sum()
mask_SNO = df1[(df1['pUnit'].isnull())]
           .drop(['bUnit','pUnit','Regional'],axis='columns')
df1_SNO = df1_SNO.append(mask_SNO)
df1_SNO['Date'] = pd.to_datetime(df1_SNO['Date'])
df1_SNO = df1_SNO.set_index('Date').sort_index(axis = 0)

idx = pd.date_range(start = df1_SNO.index.min(),
                    end = df1_SNO.index.max(), freq='D')
df1_SNO = df1_SNO.reindex(idx)

df1_SNO[df1_SNO['Deliveries'] < 0] = 0
df1_SNO = df1_SNO.fillna(0)

# Criando dataframes para Rondonópolis

df1_RNO = df1[(df1['pUnit'] == 'RNO')]

dm1_RNO = df1_RNO.copy()
dm1_RNO['Date'] = pd.to_datetime(df1_RNO['Date'])
dm1_RNO = df1_RNO.groupby(pd.Grouper(key='Date', freq='M'))['Deliveries'].sum()
#dm.index = dm.index.strftime('%B')
dm1_RNO = pd.DataFrame(dm1_RNO)

df1_RNO = df1_RNO.groupby(by='Date',as_index=False)['Deliveries'].sum()
mask_RNO = df1[(df1['pUnit'].isnull())]
           .drop(['bUnit','pUnit','Regional'],axis='columns')
df1_RNO = df1_RNO.append(mask_RNO)
df1_RNO['Date'] = pd.to_datetime(df1_RNO['Date'])
df1_RNO = df1_RNO.set_index('Date').sort_index(axis = 0)

idx = pd.date_range(start = df1_RNO.index.min(),
                    end = df1_RNO.index.max(), freq='D')
df1_RNO = df1_RNO.reindex(idx)

df1_RNO[df1_RNO['Deliveries'] < 0] = 0
```

```
df1_RNO = df1_RNO.fillna(0)

# Criando dataframes para Catalão

df1_CTO = df1[(df1['pUnit'] == 'CTO')]

dm1_CTO = df1_CTO.copy()
dm1_CTO['Date'] = pd.to_datetime(df1_CTO['Date'])
dm1_CTO = df1_CTO.groupby(pd.Grouper(key='Date', freq='M'))['Deliveries'].sum()
#dm.index = dm.index.strftime('%B')
dm1_CTO = pd.DataFrame(dm1_CTO)

df1_CTO = df1_CTO.groupby(by='Date',as_index=False)['Deliveries'].sum()
mask_CTO = df1[(df1['pUnit'].isnull())]
           .drop(['bUnit','pUnit','Regional'],axis='columns')
df1_CTO = df1_CTO.append(mask_CTO)
df1_CTO['Date'] = pd.to_datetime(df1_CTO['Date'])
df1_CTO = df1_CTO.set_index('Date').sort_index(axis = 0)

idx = pd.date_range(start = df1_CTO.index.min(),
                    end = df1_CTO.index.max(), freq='D')
df1_CTO = df1_CTO.reindex(idx)

df1_CTO[df1_CTO['Deliveries'] < 0] = 0
df1_CTO = df1_CTO.fillna(0)

# Criando dataframes para Paranaguá

df1_PGU = df1[(df1['pUnit'] == 'PGU')]

dm1_PGU = df1_PGU.copy()
dm1_PGU['Date'] = pd.to_datetime(df1_PGU['Date'])
dm1_PGU = df1_PGU.groupby(pd.Grouper(key='Date', freq='M'))['Deliveries'].sum()
#dm.index = dm.index.strftime('%B')
dm1_PGU = pd.DataFrame(dm1_PGU)

df1_PGU = df1_PGU.groupby(by='Date',as_index=False)['Deliveries'].sum()
mask_PGU = df1[(df1['pUnit'].isnull())]
           .drop(['bUnit','pUnit','Regional'],axis='columns')
```

```
df1_PGU = df1_PGU.append(mask_PGU)
df1_PGU['Date'] = pd.to_datetime(df1_PGU['Date'])
df1_PGU = df1_PGU.set_index('Date').sort_index(axis = 0)

idx = pd.date_range(start = df1_PGU.index.min(),
                    end = df1_PGU.index.max(), freq='D')
df1_PGU = df1_PGU.reindex(idx)

df1_PGU[df1_PGU['Deliveries'] < 0] = 0
df1_PGU = df1_PGU.fillna(0)

# Criando dataframes para Araguari

df1_ARO = df1[(df1['pUnit'] == 'ARO')]

dm1_ARO = df1_ARO.copy()
dm1_ARO['Date'] = pd.to_datetime(df1_ARO['Date'])
dm1_ARO = df1_ARO.groupby(pd.Grouper(key='Date', freq='M'))['Deliveries'].sum()
#dm.index = dm.index.strftime('%B')
dm1_ARO = pd.DataFrame(dm1_ARO)

df1_ARO = df1_ARO.groupby(by='Date',as_index=False)['Deliveries'].sum()
mask_ARO = df1[(df1['pUnit'].isnull())]
           .drop(['bUnit','pUnit','Regional'],axis='columns')
df1_ARO = df1_ARO.append(mask_ARO)
df1_ARO['Date'] = pd.to_datetime(df1_ARO['Date'])
df1_ARO = df1_ARO.set_index('Date').sort_index(axis = 0)

idx = pd.date_range(start = df1_ARO.index.min(),
                    end = df1_ARO.index.max(), freq='D')
df1_ARO = df1_ARO.reindex(idx)

df1_ARO[df1_ARO['Deliveries'] < 0] = 0
df1_ARO = df1_ARO.fillna(0)

## Definir quais são as Matérias-Primas de interesse
MP1 = 'KCL'
MP2 = 'UREIA'
MP3 = 'MAP'
```

```
MP4 = 'TSP'
MP5 = 'SSP'

# CSV_prodution.csv
# Fazendo tratamento da base de produção (consumo de matéria-prima)
para criar DataFrames para cada unidade produtora


# Criando DataFrames para Porto Nacional
df2_PNO = df2[(df2['pUnit'] == 'PNO')]
df2_PNO_MP1 = df2_PNO[(df2_PNO['ProductGenDesc'] == MP1)]
df2_PNO_MP2 = df2_PNO[(df2_PNO['ProductGenDesc'] == MP2)]
df2_PNO_MP3 = df2_PNO[(df2_PNO['ProductGenDesc'] == MP3)]
df2_PNO_MP4 = df2_PNO[(df2_PNO['ProductGenDesc'] == MP4)]
df2_PNO_MP5 = df2_PNO[(df2_PNO['ProductGenDesc'] == MP5)]

dm2_PNO_MP1 = df2_PNO_MP1.copy()
dm2_PNO_MP1 = dm2_PNO_MP1.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_PNO_MP1 = pd.DataFrame(dm2_PNO_MP1)
df2_PNO_MP1 = df2_PNO_MP1.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_PNO_MP1 = pd.DataFrame(df2_PNO_MP1)

dm2_PNO_MP2 = df2_PNO_MP2.copy()
dm2_PNO_MP2 = dm2_PNO_MP2.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_PNO_MP2 = pd.DataFrame(dm2_PNO_MP2)
df2_PNO_MP2 = df2_PNO_MP2.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_PNO_MP2 = pd.DataFrame(df2_PNO_MP2)

dm2_PNO_MP3 = df2_PNO_MP3.copy()
dm2_PNO_MP3 = dm2_PNO_MP3.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_PNO_MP3 = pd.DataFrame(dm2_PNO_MP3)
df2_PNO_MP3 = df2_PNO_MP3.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_PNO_MP3 = pd.DataFrame(df2_PNO_MP3)
```

```
dm2_PNO_MP4 = df2_PNO_MP4.copy()
dm2_PNO_MP4 = dm2_PNO_MP4.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_PNO_MP4 = pd.DataFrame(dm2_PNO_MP4)
df2_PNO_MP4 = df2_PNO_MP4.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_PNO_MP4 = pd.DataFrame(df2_PNO_MP4)

dm2_PNO_MP5 = df2_PNO_MP5.copy()
dm2_PNO_MP5 = dm2_PNO_MP5.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_PNO_MP5 = pd.DataFrame(dm2_PNO_MP5)
df2_PNO_MP5 = df2_PNO_MP5.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_PNO_MP5 = pd.DataFrame(df2_PNO_MP5)

# Criando DataFrames para São Luís
df2_SLO = df2[(df2['pUnit'] == 'SLO')]
df2_SLO_MP1 = df2_SLO[(df2_SLO['ProductGenDesc'] == MP1)]
df2_SLO_MP2 = df2_SLO[(df2_SLO['ProductGenDesc'] == MP2)]
df2_SLO_MP3 = df2_SLO[(df2_SLO['ProductGenDesc'] == MP3)]
df2_SLO_MP4 = df2_SLO[(df2_SLO['ProductGenDesc'] == MP4)]
df2_SLO_MP5 = df2_SLO[(df2_SLO['ProductGenDesc'] == MP5)]

dm2_SLO_MP1 = df2_SLO_MP1.copy()
dm2_SLO_MP1 = dm2_SLO_MP1.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_SLO_MP1 = pd.DataFrame(dm2_SLO_MP1)
df2_SLO_MP1 = df2_SLO_MP1.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_SLO_MP1 = pd.DataFrame(df2_SLO_MP1)

dm2_SLO_MP2 = df2_SLO_MP2.copy()
dm2_SLO_MP2 = dm2_SLO_MP2.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_SLO_MP2 = pd.DataFrame(dm2_SLO_MP2)
df2_SLO_MP2 = df2_SLO_MP2.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
```

```
df2_SLO_MP2 = pd.DataFrame(df2_SLO_MP2)

dm2_SLO_MP3 = df2_SLO_MP3.copy()
dm2_SLO_MP3 = dm2_SLO_MP3.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_SLO_MP3 = pd.DataFrame(dm2_SLO_MP3)
df2_SLO_MP3 = df2_SLO_MP3.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_SLO_MP3 = pd.DataFrame(df2_SLO_MP3)

dm2_SLO_MP4 = df2_SLO_MP4.copy()
dm2_SLO_MP4 = dm2_SLO_MP4.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_SLO_MP4 = pd.DataFrame(dm2_SLO_MP4)
df2_SLO_MP4 = df2_SLO_MP4.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_SLO_MP4 = pd.DataFrame(df2_SLO_MP4)

dm2_SLO_MP5 = df2_SLO_MP5.copy()
dm2_SLO_MP5 = dm2_SLO_MP5.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_SLO_MP5 = pd.DataFrame(dm2_SLO_MP5)
df2_SLO_MP5 = df2_SLO_MP5.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_SLO_MP5 = pd.DataFrame(df2_SLO_MP5)

# Criando DataFrames para Querência
df2_QRO = df2[(df2['pUnit'] == 'QRO')]
df2_QRO_MP1 = df2_QRO[(df2_QRO['ProductGenDesc'] == MP1)]
df2_QRO_MP2 = df2_QRO[(df2_QRO['ProductGenDesc'] == MP2)]
df2_QRO_MP3 = df2_QRO[(df2_QRO['ProductGenDesc'] == MP3)]
df2_QRO_MP4 = df2_QRO[(df2_QRO['ProductGenDesc'] == MP4)]
df2_QRO_MP5 = df2_QRO[(df2_QRO['ProductGenDesc'] == MP5)]

dm2_QRO_MP1 = df2_QRO_MP1.copy()
dm2_QRO_MP1 = dm2_QRO_MP1.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_QRO_MP1 = pd.DataFrame(dm2_QRO_MP1)
df2_QRO_MP1 = df2_QRO_MP1.groupby(pd.Grouper(key='Date',
```

```
                                                freq='D'))['Quantity'].sum()
df2_QRO_MP1 = pd.DataFrame(df2_QRO_MP1)

dm2_QRO_MP2 = df2_QRO_MP2.copy()
dm2_QRO_MP2 = dm2_QRO_MP2.groupby(pd.Grouper(key='Date',
                                                freq='M'))['Quantity'].sum()
dm2_QRO_MP2 = pd.DataFrame(dm2_QRO_MP2)
df2_QRO_MP2 = df2_QRO_MP2.groupby(pd.Grouper(key='Date',
                                                freq='D'))['Quantity'].sum()
df2_QRO_MP2 = pd.DataFrame(df2_QRO_MP2)

dm2_QRO_MP3 = df2_QRO_MP3.copy()
dm2_QRO_MP3 = dm2_QRO_MP3.groupby(pd.Grouper(key='Date',
                                                freq='M'))['Quantity'].sum()
dm2_QRO_MP3 = pd.DataFrame(dm2_QRO_MP3)
df2_QRO_MP3 = df2_QRO_MP3.groupby(pd.Grouper(key='Date',
                                                freq='D'))['Quantity'].sum()
df2_QRO_MP3 = pd.DataFrame(df2_QRO_MP3)

dm2_QRO_MP4 = df2_QRO_MP4.copy()
dm2_QRO_MP4 = dm2_QRO_MP4.groupby(pd.Grouper(key='Date',
                                                freq='M'))['Quantity'].sum()
dm2_QRO_MP4 = pd.DataFrame(dm2_QRO_MP4)
df2_QRO_MP4 = df2_QRO_MP4.groupby(pd.Grouper(key='Date',
                                                freq='D'))['Quantity'].sum()
df2_QRO_MP4 = pd.DataFrame(df2_QRO_MP4)

dm2_QRO_MP5 = df2_QRO_MP5.copy()
dm2_QRO_MP5 = dm2_QRO_MP5.groupby(pd.Grouper(key='Date',
                                                freq='M'))['Quantity'].sum()
dm2_QRO_MP5 = pd.DataFrame(dm2_QRO_MP5)
df2_QRO_MP5 = df2_QRO_MP5.groupby(pd.Grouper(key='Date',
                                                freq='D'))['Quantity'].sum()
df2_QRO_MP5 = pd.DataFrame(df2_QRO_MP5)

# Criando DataFrames para Barcarena
df2_BCO = df2[(df2['pUnit'] == 'BCO')]
df2_BCO_MP1 = df2_BCO[(df2_BCO['ProductGenDesc'] == MP1)]
df2_BCO_MP2 = df2_BCO[(df2_BCO['ProductGenDesc'] == MP2)]
```

```
df2_BCO_MP3 = df2_BCO[(df2_BCO['ProductGenDesc'] == MP3)]
df2_BCO_MP4 = df2_BCO[(df2_BCO['ProductGenDesc'] == MP4)]
df2_BCO_MP5 = df2_BCO[(df2_BCO['ProductGenDesc'] == MP5)]

dm2_BCO_MP1 = df2_BCO_MP1.copy()
dm2_BCO_MP1 = dm2_BCO_MP1.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_BCO_MP1 = pd.DataFrame(dm2_BCO_MP1)
df2_BCO_MP1 = df2_BCO_MP1.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_BCO_MP1 = pd.DataFrame(df2_BCO_MP1)

dm2_BCO_MP2 = df2_BCO_MP2.copy()
dm2_BCO_MP2 = dm2_BCO_MP2.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_BCO_MP2 = pd.DataFrame(dm2_BCO_MP2)
df2_BCO_MP2 = df2_BCO_MP2.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_BCO_MP2 = pd.DataFrame(df2_BCO_MP2)

dm2_BCO_MP3 = df2_BCO_MP3.copy()
dm2_BCO_MP3 = dm2_BCO_MP3.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_BCO_MP3 = pd.DataFrame(dm2_BCO_MP3)
df2_BCO_MP3 = df2_BCO_MP3.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_BCO_MP3 = pd.DataFrame(df2_BCO_MP3)

dm2_BCO_MP4 = df2_BCO_MP4.copy()
dm2_BCO_MP4 = dm2_BCO_MP4.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_BCO_MP4 = pd.DataFrame(dm2_BCO_MP4)
df2_BCO_MP4 = df2_BCO_MP4.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_BCO_MP4 = pd.DataFrame(df2_BCO_MP4)

dm2_BCO_MP5 = df2_BCO_MP5.copy()
dm2_BCO_MP5 = dm2_BCO_MP5.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
```

```
dm2_BCO_MP5 = pd.DataFrame(dm2_BCO_MP5)
df2_BCO_MP5 = df2_BCO_MP5.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_BCO_MP5 = pd.DataFrame(df2_BCO_MP5)

# Criando DataFrames para Fertimaxi
df2_FTX = df2[(df2['pUnit'] == 'FTX')]
df2_FTX_MP1 = df2_FTX[(df2_FTX['ProductGenDesc'] == MP1)]
df2_FTX_MP2 = df2_FTX[(df2_FTX['ProductGenDesc'] == MP2)]
df2_FTX_MP3 = df2_FTX[(df2_FTX['ProductGenDesc'] == MP3)]
df2_FTX_MP4 = df2_FTX[(df2_FTX['ProductGenDesc'] == MP4)]
df2_FTX_MP5 = df2_FTX[(df2_FTX['ProductGenDesc'] == MP5)]

dm2_FTX_MP1 = df2_FTX_MP1.copy()
dm2_FTX_MP1 = dm2_FTX_MP1.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_FTX_MP1 = pd.DataFrame(dm2_FTX_MP1)
df2_FTX_MP1 = df2_FTX_MP1.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_FTX_MP1 = pd.DataFrame(df2_FTX_MP1)

dm2_FTX_MP2 = df2_FTX_MP2.copy()
dm2_FTX_MP2 = dm2_FTX_MP2.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_FTX_MP2 = pd.DataFrame(dm2_FTX_MP2)
df2_FTX_MP2 = df2_FTX_MP2.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_FTX_MP2 = pd.DataFrame(df2_FTX_MP2)

dm2_FTX_MP3 = df2_FTX_MP3.copy()
dm2_FTX_MP3 = dm2_FTX_MP3.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_FTX_MP3 = pd.DataFrame(dm2_FTX_MP3)
df2_FTX_MP3 = df2_FTX_MP3.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_FTX_MP3 = pd.DataFrame(df2_FTX_MP3)

dm2_FTX_MP4 = df2_FTX_MP4.copy()
dm2_FTX_MP4 = dm2_FTX_MP4.groupby(pd.Grouper(key='Date',
```

```
                                                freq='M'))['Quantity'].sum()
dm2_FTX_MP4 = pd.DataFrame(dm2_FTX_MP4)
df2_FTX_MP4 = df2_FTX_MP4.groupby(pd.Grouper(key='Date',
                                                freq='D'))['Quantity'].sum()
df2_FTX_MP4 = pd.DataFrame(df2_FTX_MP4)

dm2_FTX_MP5 = df2_FTX_MP5.copy()
dm2_FTX_MP5 = dm2_FTX_MP5.groupby(pd.Grouper(key='Date',
                                                freq='M'))['Quantity'].sum()
dm2_FTX_MP5 = pd.DataFrame(dm2_FTX_MP5)
df2_FTX_MP5 = df2_FTX_MP5.groupby(pd.Grouper(key='Date',
                                                freq='D'))['Quantity'].sum()
df2_FTX_MP5 = pd.DataFrame(df2_FTX_MP5)

# Criando DataFrames para Intermarítima
df2_INT = df2[(df2['pUnit'] == 'INT')]
df2_INT_MP1 = df2_INT[(df2_INT['ProductGenDesc'] == MP1)]
df2_INT_MP2 = df2_INT[(df2_INT['ProductGenDesc'] == MP2)]
df2_INT_MP3 = df2_INT[(df2_INT['ProductGenDesc'] == MP3)]
df2_INT_MP4 = df2_INT[(df2_INT['ProductGenDesc'] == MP4)]
df2_INT_MP5 = df2_INT[(df2_INT['ProductGenDesc'] == MP5)]

dm2_INT_MP1 = df2_INT_MP1.copy()
dm2_INT_MP1 = dm2_INT_MP1.groupby(pd.Grouper(key='Date',
                                                freq='M'))['Quantity'].sum()
dm2_INT_MP1 = pd.DataFrame(dm2_INT_MP1)
df2_INT_MP1 = df2_INT_MP1.groupby(pd.Grouper(key='Date',
                                                freq='D'))['Quantity'].sum()
df2_INT_MP1 = pd.DataFrame(df2_INT_MP1)

dm2_INT_MP2 = df2_INT_MP2.copy()
dm2_INT_MP2 = dm2_INT_MP2.groupby(pd.Grouper(key='Date',
                                                freq='M'))['Quantity'].sum()
dm2_INT_MP2 = pd.DataFrame(dm2_INT_MP2)
df2_INT_MP2 = df2_INT_MP2.groupby(pd.Grouper(key='Date',
                                                freq='D'))['Quantity'].sum()
df2_INT_MP2 = pd.DataFrame(df2_INT_MP2)

dm2_INT_MP3 = df2_INT_MP3.copy()
```

```
dm2_INT_MP3 = dm2_INT_MP3.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_INT_MP3 = pd.DataFrame(dm2_INT_MP3)
df2_INT_MP3 = df2_INT_MP3.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_INT_MP3 = pd.DataFrame(df2_INT_MP3)

dm2_INT_MP4 = df2_INT_MP4.copy()
dm2_INT_MP4 = dm2_INT_MP4.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_INT_MP4 = pd.DataFrame(dm2_INT_MP4)
df2_INT_MP4 = df2_INT_MP4.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_INT_MP4 = pd.DataFrame(df2_INT_MP4)

dm2_INT_MP5 = df2_INT_MP5.copy()
dm2_INT_MP5 = dm2_INT_MP5.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_INT_MP5 = pd.DataFrame(dm2_INT_MP5)
df2_INT_MP5 = df2_INT_MP5.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_INT_MP5 = pd.DataFrame(df2_INT_MP5)

# Criando DataFrames para Sinop
df2_SNO = df2[(df2['pUnit'] == 'SNO')]
df2_SNO_MP1 = df2_SNO[(df2_SNO['ProductGenDesc'] == MP1)]
df2_SNO_MP2 = df2_SNO[(df2_SNO['ProductGenDesc'] == MP2)]
df2_SNO_MP3 = df2_SNO[(df2_SNO['ProductGenDesc'] == MP3)]
df2_SNO_MP4 = df2_SNO[(df2_SNO['ProductGenDesc'] == MP4)]
df2_SNO_MP5 = df2_SNO[(df2_SNO['ProductGenDesc'] == MP5)]

dm2_SNO_MP1 = df2_SNO_MP1.copy()
dm2_SNO_MP1 = dm2_SNO_MP1.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_SNO_MP1 = pd.DataFrame(dm2_SNO_MP1)
df2_SNO_MP1 = df2_SNO_MP1.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_SNO_MP1 = pd.DataFrame(df2_SNO_MP1)
```

```
dm2_SNO_MP2 = df2_SNO_MP2.copy()
dm2_SNO_MP2 = dm2_SNO_MP2.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_SNO_MP2 = pd.DataFrame(dm2_SNO_MP2)
df2_SNO_MP2 = df2_SNO_MP2.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_SNO_MP2 = pd.DataFrame(df2_SNO_MP2)

dm2_SNO_MP3 = df2_SNO_MP3.copy()
dm2_SNO_MP3 = dm2_SNO_MP3.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_SNO_MP3 = pd.DataFrame(dm2_SNO_MP3)
df2_SNO_MP3 = df2_SNO_MP3.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_SNO_MP3 = pd.DataFrame(df2_SNO_MP3)

dm2_SNO_MP4 = df2_SNO_MP4.copy()
dm2_SNO_MP4 = dm2_SNO_MP4.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_SNO_MP4 = pd.DataFrame(dm2_SNO_MP4)
df2_SNO_MP4 = df2_SNO_MP4.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_SNO_MP4 = pd.DataFrame(df2_SNO_MP4)

dm2_SNO_MP5 = df2_SNO_MP5.copy()
dm2_SNO_MP5 = dm2_SNO_MP5.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_SNO_MP5 = pd.DataFrame(dm2_SNO_MP5)
df2_SNO_MP5 = df2_SNO_MP5.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_SNO_MP5 = pd.DataFrame(df2_SNO_MP5)

# Criando DataFrames para Rondonópolis
df2_RNO = df2[(df2['pUnit'] == 'RNO')]
df2_RNO_MP1 = df2_RNO[(df2_RNO['ProductGenDesc'] == MP1)]
df2_RNO_MP2 = df2_RNO[(df2_RNO['ProductGenDesc'] == MP2)]
df2_RNO_MP3 = df2_RNO[(df2_RNO['ProductGenDesc'] == MP3)]
df2_RNO_MP4 = df2_RNO[(df2_RNO['ProductGenDesc'] == MP4)]
df2_RNO_MP5 = df2_RNO[(df2_RNO['ProductGenDesc'] == MP5)]
```

```
dm2_RNO_MP1 = df2_RNO_MP1.copy()
dm2_RNO_MP1 = dm2_RNO_MP1.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_RNO_MP1 = pd.DataFrame(dm2_RNO_MP1)
df2_RNO_MP1 = df2_RNO_MP1.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_RNO_MP1 = pd.DataFrame(df2_RNO_MP1)

dm2_RNO_MP2 = df2_RNO_MP2.copy()
dm2_RNO_MP2 = dm2_RNO_MP2.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_RNO_MP2 = pd.DataFrame(dm2_RNO_MP2)
df2_RNO_MP2 = df2_RNO_MP2.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_RNO_MP2 = pd.DataFrame(df2_RNO_MP2)

dm2_RNO_MP3 = df2_RNO_MP3.copy()
dm2_RNO_MP3 = dm2_RNO_MP3.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_RNO_MP3 = pd.DataFrame(dm2_RNO_MP3)
df2_RNO_MP3 = df2_RNO_MP3.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_RNO_MP3 = pd.DataFrame(df2_RNO_MP3)

dm2_RNO_MP4 = df2_RNO_MP4.copy()
dm2_RNO_MP4 = dm2_RNO_MP4.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_RNO_MP4 = pd.DataFrame(dm2_RNO_MP4)
df2_RNO_MP4 = df2_RNO_MP4.groupby(pd.Grouper(key='Date',
                                            freq='D'))['Quantity'].sum()
df2_RNO_MP4 = pd.DataFram e(df2_RNO_MP4)

dm2_RNO_MP5 = df2_RNO_MP5.copy()
dm2_RNO_MP5 = dm2_RNO_MP5.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_RNO_MP5 = pd.DataFrame(dm2_RNO_MP5)
df2_RNO_MP5 = df2_RNO_MP5.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
```

```
df2_RNO_MP5 = pd.DataFrame(df2_RNO_MP5)

# Criando DataFrames para Catalão
df2_CTO = df2[(df2['pUnit'] == 'CTO')]
df2_CTO_MP1 = df2_CTO[(df2_CTO['ProductGenDesc'] == MP1)]
df2_CTO_MP2 = df2_CTO[(df2_CTO['ProductGenDesc'] == MP2)]
df2_CTO_MP3 = df2_CTO[(df2_CTO['ProductGenDesc'] == MP3)]
df2_CTO_MP4 = df2_CTO[(df2_CTO['ProductGenDesc'] == MP4)]
df2_CTO_MP5 = df2_CTO[(df2_CTO['ProductGenDesc'] == MP5)]

dm2_CTO_MP1 = df2_CTO_MP1.copy()
dm2_CTO_MP1 = dm2_CTO_MP1.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_CTO_MP1 = pd.DataFrame(dm2_CTO_MP1)
df2_CTO_MP1 = df2_CTO_MP1.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_CTO_MP1 = pd.DataFrame(df2_CTO_MP1)

dm2_CTO_MP2 = df2_CTO_MP2.copy()
dm2_CTO_MP2 = dm2_CTO_MP2.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_CTO_MP2 = pd.DataFrame(dm2_CTO_MP2)
df2_CTO_MP2 = df2_CTO_MP2.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_CTO_MP2 = pd.DataFrame(df2_CTO_MP2)

dm2_CTO_MP3 = df2_CTO_MP3.copy()
dm2_CTO_MP3 = dm2_CTO_MP3.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_CTO_MP3 = pd.DataFrame(dm2_CTO_MP3)
df2_CTO_MP3 = df2_CTO_MP3.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_CTO_MP3 = pd.DataFrame(df2_CTO_MP3)

dm2_CTO_MP4 = df2_CTO_MP4.copy()
dm2_CTO_MP4 = dm2_CTO_MP4.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_CTO_MP4 = pd.DataFrame(dm2_CTO_MP4)
df2_CTO_MP4 = df2_CTO_MP4.groupby(pd.Grouper(key='Date',
```

```
                                                freq='D'))['Quantity'].sum()
df2_CTO_MP4 = pd.DataFrame(df2_CTO_MP4)

dm2_CTO_MP5 = df2_CTO_MP5.copy()
dm2_CTO_MP5 = dm2_CTO_MP5.groupby(pd.Grouper(key='Date',
                                                freq='M'))['Quantity'].sum()
dm2_CTO_MP5 = pd.DataFrame(dm2_CTO_MP5)
df2_CTO_MP5 = df2_CTO_MP5.groupby(pd.Grouper(key='Date',
                                                freq='D'))['Quantity'].sum()
df2_CTO_MP5 = pd.DataFrame(df2_CTO_MP5)

# Criando DataFrames para Paranaguá
df2_PGU = df2[(df2['pUnit'] == 'PGU')]
df2_PGU_MP1 = df2_PGU[(df2_PGU['ProductGenDesc'] == MP1)]
df2_PGU_MP2 = df2_PGU[(df2_PGU['ProductGenDesc'] == MP2)]
df2_PGU_MP3 = df2_PGU[(df2_PGU['ProductGenDesc'] == MP3)]
df2_PGU_MP4 = df2_PGU[(df2_PGU['ProductGenDesc'] == MP4)]
df2_PGU_MP5 = df2_PGU[(df2_PGU['ProductGenDesc'] == MP5)]

dm2_PGU_MP1 = df2_PGU_MP1.copy()
dm2_PGU_MP1 = dm2_PGU_MP1.groupby(pd.Grouper(key='Date',
                                                freq='M'))['Quantity'].sum()
dm2_PGU_MP1 = pd.DataFrame(dm2_PGU_MP1)
df2_PGU_MP1 = df2_PGU_MP1.groupby(pd.Grouper(key='Date',
                                                freq='D'))['Quantity'].sum()
df2_PGU_MP1 = pd.DataFrame(df2_PGU_MP1)

dm2_PGU_MP2 = df2_PGU_MP2.copy()
dm2_PGU_MP2 = dm2_PGU_MP2.groupby(pd.Grouper(key='Date',
                                                freq='M'))['Quantity'].sum()
dm2_PGU_MP2 = pd.DataFrame(dm2_PGU_MP2)
df2_PGU_MP2 = df2_PGU_MP2.groupby(pd.Grouper(key='Date',
                                                freq='D'))['Quantity'].sum()
df2_PGU_MP2 = pd.DataFrame(df2_PGU_MP2)

dm2_PGU_MP3 = df2_PGU_MP3.copy()
dm2_PGU_MP3 = dm2_PGU_MP3.groupby(pd.Grouper(key='Date',
                                                freq='M'))['Quantity'].sum()
dm2_PGU_MP3 = pd.DataFrame(dm2_PGU_MP3)
```

```
df2_PGU_MP3 = df2_PGU_MP3.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_PGU_MP3 = pd.DataFrame(df2_PGU_MP3)

dm2_PGU_MP4 = df2_PGU_MP4.copy()
dm2_PGU_MP4 = dm2_PGU_MP4.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_PGU_MP4 = pd.DataFrame(dm2_PGU_MP4)
df2_PGU_MP4 = df2_PGU_MP4.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_PGU_MP4 = pd.DataFrame(df2_PGU_MP4)

dm2_PGU_MP5 = df2_PGU_MP5.copy()
dm2_PGU_MP5 = dm2_PGU_MP5.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_PGU_MP5 = pd.DataFrame(dm2_PGU_MP5)
df2_PGU_MP5 = df2_PGU_MP5.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_PGU_MP5 = pd.DataFrame(df2_PGU_MP5)

# Criando DataFrames para Araguari
df2_ARO = df2[(df2['pUnit'] == 'ARO')]
df2_ARO_MP1 = df2_ARO[(df2_ARO['ProductGenDesc'] == MP1)]
df2_ARO_MP2 = df2_ARO[(df2_ARO['ProductGenDesc'] == MP2)]
df2_ARO_MP3 = df2_ARO[(df2_ARO['ProductGenDesc'] == MP3)]
df2_ARO_MP4 = df2_ARO[(df2_ARO['ProductGenDesc'] == MP4)]
df2_ARO_MP5 = df2_ARO[(df2_ARO['ProductGenDesc'] == MP5)]

dm2_ARO_MP1 = df2_ARO_MP1.copy()
dm2_ARO_MP1 = dm2_ARO_MP1.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_ARO_MP1 = pd.DataFrame(dm2_ARO_MP1)
df2_ARO_MP1 = df2_ARO_MP1.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_ARO_MP1 = pd.DataFrame(df2_ARO_MP1)

dm2_ARO_MP2 = df2_ARO_MP2.copy()
dm2_ARO_MP2 = dm2_ARO_MP2.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
```

```
dm2_ARO_MP2 = pd.DataFrame(dm2_ARO_MP2)
df2_ARO_MP2 = df2_ARO_MP2.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_ARO_MP2 = pd.DataFrame(df2_ARO_MP2)

dm2_ARO_MP3 = df2_ARO_MP3.copy()
dm2_ARO_MP3 = dm2_ARO_MP3.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_ARO_MP3 = pd.DataFrame(dm2_ARO_MP3)
df2_ARO_MP3 = df2_ARO_MP3.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_ARO_MP3 = pd.DataFrame(df2_ARO_MP3)

dm2_ARO_MP4 = df2_ARO_MP4.copy()
dm2_ARO_MP4 = dm2_ARO_MP4.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_ARO_MP4 = pd.DataFrame(dm2_ARO_MP4)
df2_ARO_MP4 = df2_ARO_MP4.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_ARO_MP4 = pd.DataFrame(df2_ARO_MP4)

dm2_ARO_MP5 = df2_ARO_MP5.copy()
dm2_ARO_MP5 = dm2_ARO_MP5.groupby(pd.Grouper(key='Date',
                                             freq='M'))['Quantity'].sum()
dm2_ARO_MP5 = pd.DataFrame(dm2_ARO_MP5)
df2_ARO_MP5 = df2_ARO_MP5.groupby(pd.Grouper(key='Date',
                                             freq='D'))['Quantity'].sum()
df2_ARO_MP5 = pd.DataFrame(df2_ARO_MP5)
```