UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

ESCOLA DE ENGENHARIA

DEPARTAMENTO DE ENGENHARIA QUÍMICA

TRABALHO DE CONCLUSÃO DE CURSO EM ENGENHARIA QUÍMICA

Modelos Estocásticos e Redes Neurais: Estudo Comparativo para Previsão de Séries Temporais

Autor: Gustavo de Oliveira e Souza

Orientador: Jorge Otávio Trierweiler

Porto Alegre, julho de 2019.

AGRADECIMENTOS

Registro meus agradecimentos aos meus pais, Marcos Antonio de Almeida Souza e Vera Lucia de Oliveira, por estarem sempre ao meu lado nos bons momentos e por sempre terem me apoiado e me aconselhado nos momentos mais difíceis. Ainda, por terem fornecido a mim e ao meu irmão as condições de buscar sempre a excelência, porém sem nunca nos esquecermos dos nossos valores.

Agradeço à minha avó, Lair Rebello de Souza, pelo apoio a minha criação, e pelo amor incondicional, mesmo que distante. Também, ao meu irmão, Daniel de Oliveira e Souza, por ser meu segundo exemplo de engenheiro, que exerce com amor e altruísmo sua profissão.

Agradeço do fundo do meu coração à Kethleen Krawcyk Corneli, por crescer junto a mim e estar sempre ao meu lado, tanto nos momentos felizes quanto frente a dificuldades.

Registro meu agradecimento ao Prof. Dr. Jorge Otávio Trierweiler pela orientação na confecção deste trabalho e por todos ensinamentos ao longo da graduação, bem como ao mestre Leonardo Mandler De Marco, pela disposição a sanar minhas dúvidas e auxiliar meu progresso. Agradeço também a todo corpo docente da UFRGS, não só por exigir sempre o melhor de mim como aluno, mas por oferecer sempre o melhor na formação de seus discípulos.

Por último, mas não menos importante, agradeço ao povo brasileiro, que, com muita luta pôde tornar realidade o sonho – meu e de muitos – da graduação em uma universidade do tamanho e prestígio da UFRGS. A ele, prometo retribuir diariamente da melhor maneira possível.

RESUMO

O estudo de séries temporais é extremamente útil para a análise empírica de um fenômeno e seu comportamento dinâmico. Com o desenvolvimento de novas metodologias, como ferramentas de *machine learning*, a sua aplicação cresce cada vez mais em diversas áreas de estudo. O presente trabalho procurou comparar o desempenho de modelos estocásticos, ferramentas consagradas no estudo e previsão de séries temporais, com o de modelos obtidos utilizando-se redes neurais recorrentes LSTM (*long short term memory* - memória de curto e longo prazo) para dois casos distintos: fornecimento de Gasolina A no estado do Rio Grande do Sul e degradação de uma coluna de adsorção TSA (*Temperature Swing Adsorption*) em uma unidade de desidratação de gás natural.

Para o primeiro caso, constatou-se que redes neurais LSTM superam modelos estocásticos, dos quais se sobressai o modelo SARIMA. Essa superioridade se constata em análises ao longo da última década tanto quanto para dados atuais, com os modelos de redes neurais fornecendo valores de REQM (raiz do erro quadrado médio) inferiores à 75% dos valores referentes ao modelo SARIMA para ambos os casos.

No tocante do estudo da degradação da coluna de adsorção, o trabalho abordou aplicação de redes neurais LSTM em séries distintas: a diferença de pressão na coluna em função do tempo e a resistência ao escoamento (degradação) em função do número de ciclos, o qual é representado por três séries, uma a cada etapa do ciclo de adsorção – 30%, 50% e 75%. Modelos estocásticos foram estudados na literatura para ambos os casos por Favarini (2018) e De Marco (2019), respectivamente. Para a diferença de pressão, as previsões obtidas por redes neurais LSTM foram satisfatórias, porém tanto quanto o modelo SARIMA. Já para a série da degradação, a mesma possui alta aplicabilidade em relação aos modelos estocásticos, apresentando valores de REQM inferiores aos relativos ao modelo SARIMA.

Palavras-chave: Séries temporais, SARIMA, Holt-Winters, Redes neurais LSTM.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SÍMBOLOS

t: Instante de tempo.

X_t: Variável observada.

T_t: Componente de tendência de uma série temporal.

S_t: Componente cíclica ou sazonal de uma série temporal.

ε_t: Componente residual de uma série temporal.

X'_t: Série temporal diferenciada.

s: Período sazonal.

$\hat{X}_t$: Modelo de ajuste da variável observada.

T: Conjunto discreto de pontos.

μ_t: Média aritmética.

x_i: Valores esperados da variável discreta.

$p(x_i)$: Função probabilidade.

σ_t^2: Variância.

γ_{t_1,t_2}: Autocovariância entre t_1 e t_2.

ρ_h: Autocorrelação.

h: Tamanho da defasagem.

$\hat{\rho}_h$: Autocorrelação amostral.

$\hat{\mu}$: Média das amostras.

a_h: Autocorrelação parcial.

π: Coeficiente de regressão linear relacionado ao valor anterior (Teste ADF).

B_i: Coeficiente de regressão linear relacionado às diferenciações de primeira ordem de valores defasados em *i* (Teste ADF).

m_t: Polinômio de qualquer ordem em função do tempo.

Y_t: Resíduo da tendência polinomial.

$\overline{X_t}$: Série temporal suavizada.

a: Coeficiente de suavização de nível.

d: Coeficiente de suavização de tendência.

c: Coeficiente de suavização de sazonalidade.

$\hat{T}_t$: Modelo para a Tendência da série suavizada.

$\hat{S}_t$: Modelo para a Sazonalidade da série suavizada.

$\emptyset$: Coeficiente de autoregressão.

ϵ: Erro de um modelo de ajuste para série temporal.

p: Ordem de autoregressão.

B: Operador de defasagem.

θ: Coeficiente de média móvel.

q: Ordem de média móvel.

d: Ordem de diferenciação.

P: Ordem de autoregressão sazonal.

D: Ordem de diferenciação sazonal.

Q: Ordem de média móvel sazonal.

Φ: Coeficiente de autoregressão sazonal.

Θ: Coeficiente de média móvel sazonal.

N: Número de parâmetros do modelo.

$f(y)$: Função da variável exógena *y.*

$\beta_i, i = 0,1 \ldots n.$: Coeficientes de um polinômio de regressão de ordem *n.*

h_t: Dados de entrada de uma rede neural LSTM.

C_t: Memória a longo prazo de uma rede neural LSTM.

σ: Função sigmoide.

tanh: Função tangente hiperbólica.

f_t: Portão de esquecimento (*forget gate*) de uma rede neural LSTM.

i_t: Portão de entrada (*input gate*) de uma rede neural LSTM.

o_t: Portão de saída (*output gate*) de uma rede neural LSTM.

W: Pesos de uma rede neural LSTM.

b: Ajuste por viés de uma rede neural LSTM.

Tr_t: Conjunto de treino.

Ts_t: Conjunto de teste.

fc: Tamanho das previsões.

Ne: Número de neurônios de uma rede neural LSTM.

w: *window* – Número de dados de entrada de uma rede neural LSTM.

ep: número de epócas de uma rede neural LSTM.

X_t^s: Variável observada normalizada entre 0 e 1.

R: Resistência ao escoamento na coluna de adsorção.

F: Vazão na coluna de adsorção.

ΔP: Diferença de pressão na coluna de adsorção.

ρ: Densidade do gás na coluna de adsorção.

LISTA DE ABREVIATURAS E SIGLAS

ACF – *Autocorrelation function* – Função de autocorrelação.

Adam – *Adaptive Moment Estimation* – Estimativa de momento adaptável

ADF – *Augmented Dickey-Fuller* – Dickey Fuller Aumentado.

AIC – (*Akaike's Information Criterion*) – Critério de Informação de Akaike.

ANP – Agência Nacional de Petróleo, Gás Natural e Biocombustíveis.

ARIMA – *Autoregressive-integrated-moving-average* – Autoregressivo integrado de médias móveis.

ARIMAX – *Exogenous Autoregressive-integrated-moving-average* – Autoregressivo integrado de médias móveis exógeno.

AR – *Autoregressive* – Autoregressivo.

ARMA – *Autoregressive-moving-average* – Autoregressivo de Médias móveis.

BIC – (*Bayesian Information Criterion*) – Critério de Informação Bayesiano.

C30 – Série da degradação em função do número de ciclos em uma coluna de adsorção para desidratação de GN em uma unidade TSA em 30% do ciclo de adsorção.

C50 – Série da degradação em função do número de ciclos em uma coluna de adsorção para desidratação de GN em uma unidade TSA em 50% do ciclo de adsorção.

C75 – Série da degradação em função do número de ciclos em uma coluna de adsorção para desidratação de GN em uma unidade TSA em 75% do ciclo de adsorção.

DP – Série temporal da diferença de pressão em uma coluna de adsorção para desidratação de GN em uma unidade TSA.

EQM – Erro quadrado médio.

LSE – *Least Squared Error* – Minimização dos erros quadrados.

LSTM – *Long short-term memory* – Curto e longo prazo.

PACF – *Parcial Autocorrelation function* – Função de autocorrelação parcial.

PCP – Planejamento e Controle da Produção.

SARIMA – *Seasonal Autoregressive-integrated-moving-average* – Autoregressivo integrado de médias móveis sazonal.

SARIMAX – *Exogenous Seasonal Autoregressive-integrated-moving-average* – Autoregressivo integrado de médias móveis sazonal exógeno.

SES – Suavização Exponencial Simples.

SMA – *Simple Moving Average* – Média móvel simples.

REQM – Raiz do erro quadrado médio.

REQM(%) – Raiz do erro quadrado médio percentual em relação à média da série de teste.

RNA – Redes neurais artificiais.

RL – Regressão Linear.

RNN – *Recurrent Neural Networks* – Redes Neurais Recorrentes.

TSA – *Temperature Swing Adsorption*.

VOL – Série temporal do fornecimento de Gasolina A no Rio Grande do Sul.

SUMÁRIO

1 INTRODUÇÃO

A definição de uma série temporal é o conjunto de observações de uma dada variável, cada uma realizada em um instante específico de tempo (Brockwell e Davis, 2006). De acordo com Ehlers (2009), a principal diferença entre uma série temporal e outros conjuntos de dados, como um modelo de regressão, é a relevância da ordem das observações – dada a dependência de observações vizinhas.

Neusser (2016) classifica a análise de séries temporais como parte fundamental de toda investigação empírica que visa a descrição e modelagem da evolução ao longo do tempo de uma variável ou um conjunto de variáveis de maneira estatisticamente coerente. Há diversos objetivos possíveis em se analisar séries temporais, destacando-se entre eles a descrição de propriedades da série – as quais serão abordadas detalhadamente na seção 2 deste trabalho –, controle de vida útil de equipamentos em processos industriais, estudar a relação entre duas ou mais séries temporais e predição de valores futuros (Ehlers, 2009).

Diversos métodos foram desenvolvidos a fim de se obter os melhores resultados possíveis na análise. A classe de modelos mais consagrada é a de modelos estocásticos; entretanto, recentemente, com diversos estudos sobre *machine learning* (Aprendizado de Máquina), foram desenvolvidos diversos algoritmos de previsão de séries temporais fazendo-se uso de redes neurais (Nelson, 2017).

Dentre diversas outras, algumas das áreas nas quais é possível encontrar aplicações de análises de séries temporais são ciências sociais e econômicas, ainda que sua origem seja proveniente das áreas de ciências naturais e engenharia. Alguns dos muitos possíveis exemplos de séries temporais são valores diários, mensais e anuais de, respectivamente, poluição, temperatura e precipitação atmosférica em cidades, índices diários de bolsas de valores e demanda ou venda de produtos em uma unidade de produção (Neusser, 2016; Morettin e Toloi, 2006).

Mais especificamente, pode-se destacar o mercado nacional de combustíveis fósseis como uma área de alta relevância para estudo e desenvolvimento de modelos de previsão. De acordo com a ANP (2018), a taxa prevista de aumento do consumo de derivados do petróleo entre os anos de 2016 e 2026 é superior a 19%. Dentre os mesmos, destaca-se o comércio de Gasolina A, principal componente da gasolina comum, compondo atualmente 73% da mesma, que, de 2016 a 2017, foi o único a apresentar aumento em seu percentual da matriz veicular nacional (ANP, 2018).

No tocante da produção de gás natural, a quantidade total produzida no Brasil apresentou crescimento de aproximadamente 7,8% de 2016 a 2018. Ainda, o valor acumulado em 2019 correspondente ao mês de maio supera em aproximadamente 3% o valor relativo em 2018 (ANP, 2019a). Apesar desse crescimento, os mesmos dados também descrevem queda considerável de produção de gás natural em terra, com valor aproximado de 8%. Em contrapartida, a produção em mar no período em questão apresentou uma taxa de crescimento superior a 12%.

Frente a esse cenário, o presente trabalho tem como objetivo estudar a aplicação de ambos modelos estocásticos e de redes neurais para séries temporais em dois estudos de caso distintos: o fornecimento de Gasolina A no estado do Rio Grande Sul, e o tempo de vida útil de uma coluna de adsorção em uma unidade TSA (*Temperature Swing Adsorption*) de desidratação de gás natural em uma plataforma *off-shore*.

Para o primeiro caso, o trabalho visa comparar modelos estocásticos (autoregressivos e de suavização) e redes neurais LSTM para previsões anuais de volume fornecido, avaliando os modelos pelo erro REQM (Raiz do Erro Quadrado Médio) e análise residual, além do estudo da viabilidade de aplicação dos mesmos ao longo da década. Ainda, será investigada na mesma série a dependência ou não de variáveis exógenas, escolhidas empiricamente.

Para o segundo caso, o objetivo é avaliar o potencial de aplicação do método LSTM, comparando os resultados obtidos com os de trabalhos anteriores, cuja metodologia de previsão foi a utilização de modelos estocásticos.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são abordadas as bases que fundamentam a compreensão da metodologia envolvida em análises de séries temporais. Para facilitar o entendimento o capítulo é dividido nos seguintes tópicos:

1) Estruturas de uma série temporal;
2) Processos estocásticos;
3) Modelos estocásticos;
4) Variáveis exógenas;
5) Redes neurais artificiais.

2.1 ESTRUTURAS DE UMA SÉRIE TEMPORAL

Parte fundamental da análise da série temporal de uma variável observada X_t é a seleção adequada da classe de modelos de probabilidade (Brockwell e Davis, 2006). A mesma pode ter sua estrutura interna dividida de tal maneira que grande parte de suas propriedades podem ser capturadas:

$$X_t = T_t + S_t + \varepsilon_t \qquad (2.1)$$

Sendo T_t a tendência, S_t a componente cíclica ou sazonal e ε_t os resíduos (EHLERS, 2009).

A análise de uma série temporal se dá através da modelagem de cada uma das componentes de sua estrutura, o que torna possível o compreendimento do comportamento observado da série como um todo, e por conseguinte a predição de valores futuros da mesma (Pal e Prakash, 2017).

2.1.1 Tendência

Ao exibir um comportamento crescente ou decrescente em um período significativo de tempo, ou seja, não considerando-se picos e vales da série, a série demonstra possuir uma componente de tendência. A presença de tendência em uma série temporal geralmente se deve a alterações sistêmicas do processo o qual ela representa, e é facilmente identificada através da análise visual de seu gráfico em função do tempo PAL e PRAKASH (2017).

De acordo com Morettin e Toloi (2006) e Ehlers (2009), os principais métodos para estimação da tendência são ajustes por função de tempo exponencial ou polinomial (função de ajuste mais utilizada), ou ainda curvas logística ou de Gompertz, e métodos de filtragem, ou suavização. Ainda, uma maneira eficiência de se remover a tendência de uma série temporal é através da diferenciação da mesma, ou seja, utilizando-se uma nova série X'_t, tal que que $X'_t = X_t - X_{t-1}$. Os métodos mais utilizados e de maior relevância para este trabalho serão abordados detalhadamente na seção 2.3.

2.1.2 Componente cíclica ou sazonal

A componente cíclica de uma série temporal representa movimentos sem necessariamente possuir um período de variação fixa, e que, devido a esse fator, costuma ser observada em séries temporais anuais longas ou que representam variáveis fortemente influenciadas por fatores externos instáveis.

A componente sazonal, ou sazonalidade, no entanto, representa movimentos ondulatórios repetitivos de período definido na série. A sazonalidade pode ser facilmente detectada por análise visual do gráfico da variável observada em função do tempo (Pal e Prakash, 2017)

Matematicamente, a componente sazonal pode ser explicada pela seguinte equação:

$$\dots S_{t-2s} = S_{t-s} = S_{t+s} = S_{t+2s} \dots \quad (2.2)$$

Sendo S_{t+s} e s a componente sazonal no instante *t* e o período sazonal, respectivamente (Ehlers, 2009).

Dito isso, um método mais preciso e confiável para detecção de sazonalidade é através da análise do gráfico de *autocorrelação amostral* da série, visto que séries sazonais apresentam forte autocorrelação entre defasagens (*lags*) periódicas (Morettin e Toloi, 2006).

2.1.3 Resíduos

Ehlers (2009) define os resíduos, ou ruídos, de uma série temporal como a parte não explicada da mesma. Dada essa definição, podemos concluir que o resíduo de uma série temporal pode ser obtido pela seguinte equação:

$$\varepsilon_t = X_t - \hat{X}_t \quad (2.3)$$

Sendo $\hat{X}_t$ o modelo de ajuste ou predição obtido. Idealmente, o ruído de uma série temporal será denominado *ruído branco*, ou seja, puramente aleatória, não sendo, portanto, uma função do tempo (média e variância constantes e finitas) (Brockwell e Davis, 2006).

Dados os conceitos acima, é possível afirmar que os parâmetros mencionados relacionados ao resíduo da série temporal são fundamentais para determinação da acuracidade do modelo ou da predição obtida. Conceitos específicos relacionados à caracterização dos resíduos serão abordados nas próximas seções do trabalho.

2.2 PROCESSOS ESTOCÁSTICOS

Um processo estocástico é definido como um processo determinado exclusivamente por leis probabilísticas. Ou seja, dada a seguinte equação:

$$X_t, \ \ t \in T \quad (2.4)$$

Sendo T um conjunto discreto de pontos, X_t é uma variável aleatória para cada valor de t no intervalo (Morettin e Toloi, 2006).

O melhor método de descrição para um processo estocástico é através das funções média, variância e autocovariância:

$$\mu_t = E[X(t)] = \sum_{i=1}^{\infty} x_i p(x_i) \quad (2.5)$$

$$\sigma_t^2 = Var[X(t)] = E[X^2] - \big(E(X)\big)^2 \quad (2.6)$$

$$\gamma_{t_1,t_2} = E[X(t_1) - \mu(t_1)][X(t_2) - \mu(t_2)], \ t_1, t_2 \in T \quad (2.7)$$

Sendo μ_t a média, x_i os valores esperados da variável discreta, $p(x_i)$ a função probabilidade, σ_t^2 a variância e γ_{t_1,t_2} a autocovariância entre t_1 e t_2 (Ehlers, 2009).

Ainda, de acordo com Morettin e Toloi (2006), modelos de séries temporais são descrições probabilísticas das mesmas, podendo, então, ser definidos como processos estocásticos.

2.2.1 Estacionaridade

Um processo estocástico é *estritamente estacionário* se, dado qualquer valor de τ, as características de $X_{t+\tau}$ são equivalentes às características de X_t. Em outras palavras, a distribuição de probabilidade é a mesma para ambos os casos.

Um processo pode ser classificado como *fracamente estacionário* se, para qualquer valor de *t*, se obtenha valores constantes finitos de média e variância, ou seja, $\mu_t = \mu < \infty$ e $\sigma_t^2 = \sigma^2 < \infty$. Ainda, a autocovariância de X_t e $X_{t-\tau}$ para um processo fracamente estacionário não varia com o tempo e depende apenas do tamanho da defasagem (representado por h). Matematicamente, essa condição é descrita pela seguinte equação:

$$\gamma_h \equiv Cov(X_t, X_{t-h}) = Cov\left(X_{t-\tau}, X_{t-\tau-h}\right) \qquad (2.8)$$

Dadas essas definições, é evidente que a estacionaridade dos resíduos do modelo em relação à referência será um critério crucial na avaliação da precisão do mesmo. Caso seja positiva a presença da mesma, os resíduos obtidos são classificados como *ruído branco*. Um ruído branco é, por definição, uma sequência de variáveis aleatórias com média zero, variância constante e autocovariâncias iguais a zero para todo *h* diferente de zero. (Guidolin e Pedio, 2018).

Neste trabalho, será aceita a condição de estacionaridade fraca como suficiente. Existem diversos critérios para avaliar a estacionaridade de resíduos de séries temporais e se os mesmos são, de fato, classificáveis como ruído branco. Esta seção abordará a análise de autocorrelação e o teste aumentado de Dickey-Fuller, alguns dos critérios mais consagrados para avaliação de estacionaridade.

2.2.1.1 Autocorrelação

Mencionada na seção 2.1.2 como forte indicadora de sazonalidade, a função autocorrelação (ACF, *Autocorrelation function*) é altamente utilizada como avaliação de estacionaridade de resíduos de séries temporais. A equação 2.9 descreve a função autocorrelação (ρ_h).

$$\rho_h = \frac{\gamma_h}{\gamma_0} \qquad (2.9)$$

Onde γ_h é definido pela equação 2.8 e γ_0 é o valor da autocovariância para $h = 0$, ou seja, a variância do ponto t. De acordo com Ehlers (2009), séries aleatórias não possuem correlação entre os valores defasados, ou seja, idealmente, $\rho_h = 0$ para todo $h \neq 0$.

Dado o conjunto T de observações discretas de uma variável X_t, estima-se a autocorrelação amostral através da equação 2.10.

$$\hat{\rho}_h = \frac{\sum_{t=h+1}^{T}(X_t - \hat{\mu})(X_{t-h} - \hat{\mu})}{\sum_{t=1}^{T}(X_t - \hat{\mu})^2} = \frac{\hat{\gamma}_h}{\hat{\gamma}_0} \qquad (2.10)$$

Onde $\hat{\rho}_h$ é a autocorrelação amostral $\hat{\mu}$ é a média das amostras e é obtida através da equação 2.11.

$$\hat{\mu} = \frac{1}{T}\sum_{i=t}^{T} X_t \qquad (2.11)$$

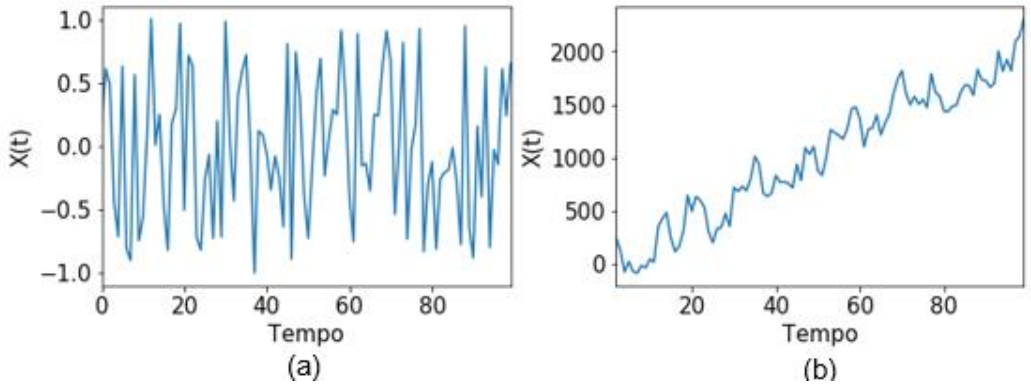
Considerando a distribuição das variáveis aleatórias idênticas com médias arbitrárias, tem-se que as autocorrelações amostrais possuem distribuição normal com média e variância tais que $\mu \approx \sigma_t^2 \approx 1/T$. Portanto, os intervalos de 95% de confiança podem ser estimados por $\pm 1{,}96\sqrt{T}$, sendo esse o critério mais utilizado para detectar a presença ou não de autocorrelação considerável (Ehlers, 2009),

Analogamente, a função autocorrelação parcial (PACF, *Partial Autocorrelation Function*) fornece a correlação da série com os seus próprios valores defasados, e é definida formalmente por:

$$a_h = Corr(X_t, X_{t-h} | X_{t-1}, \ldots, X_{t-h+1}) \qquad (2.12)$$

A análise da estacionaridade dos resíduos através das funções de autocorrelação ACF e PACF pode ser feita visualmente através de seus gráficos. A Figura 2.1(a) ilustra uma série temporal composta por pontos aleatórios com distribuição normal, enquanto a Figura 2.1(b) foi composta como uma função de seus valores defasados. Ambas foram geradas em Python utilizando a biblioteca *Numpy*, com 100 observações cada.

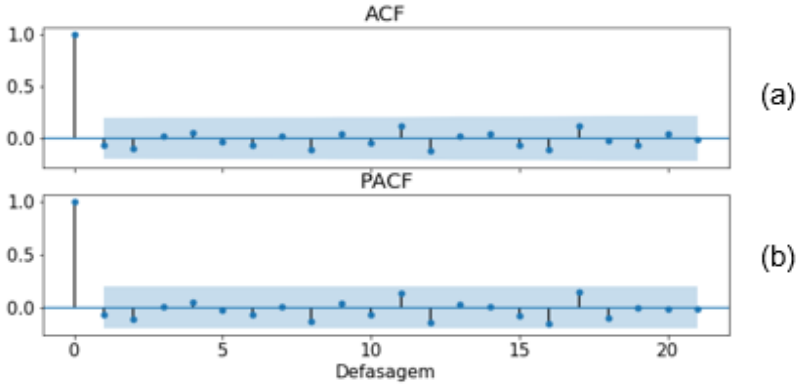
Figura 2.1 – **a)** Série aleatória com distribuição normal; **b)** Série aleatória sem distribuição normal.



Fonte: Elaborado pelo autor (2019).

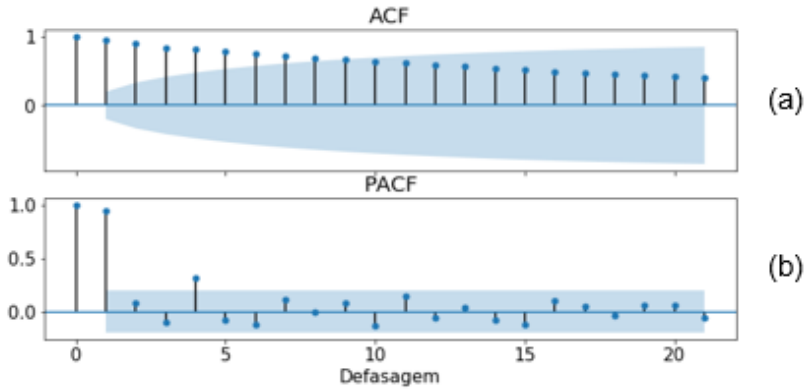
As Figuras 2.2 e 2.3 ilustram as funções autocorrelação (ACF) e autocorrelação parcial (PACF) para ambos os casos.

Figura 2.2 – Funções autocorrelação **(a)** e autocorrelação parcial **(b)** da série aleatória com distribuição normal (21 defasagens).



Fonte: Elaborado pelo autor (2019).

Figura 2.3 – Funções autocorrelação **(a)** e autocorrelação parcial **(b)** da série aleatória sem distribuição normal (21 defasagens).



Fonte: Elaborado pelo autor (2019).

A região azul dos gráficos das Figuras 2.2 e 2.3 representam o intervalo de 95% de confiança. Visto que, para a primeira série, nenhum ponto a partir de h = 1 ficou fora da região azul, a série não possui autocorrelação considerável e pode ser considerada estacionária. O mesmo não ocorre para a segunda série, cuja autocorrelação decresce gradativamente e se mantém fora da região azul até a nona defasagem.

2.2.1.2 Teste aumentado de Dickey-Fuller

Testes para detecção da necessidade de diferenciação da série temporal para obter-se estacionaridade são chamados testes de raiz unitária. Caso um processo estocástico possua raiz unitária (hipótese nula), diz-se que o mesmo é um processo não-estacionário. Logicamente, caso o teste indique presença de raiz unitária, o processo é dito estacionário.

De acordo com teste aumentado de Dickey-Fuller, ou teste ADF, a diferenciação de primeira ordem X'_t de uma série temporal da variável X_t podem ser expressas através de uma regressão linear do valor original anterior e de diferenciações de primeira ordem de h defasagens, conforme descreve a equação 2.13 (Pal e Prakash, 2017).

$$X'_t = \pi X_{t-1} + B_1 X'_{t-1} + B_2 X'_{t-2} + B_h X'_{t-h} + \varepsilon_t \qquad (2.13)$$

Sendo π e B_i, $i = \{1,2,\dots h\}$ os coeficientes de regressão linear relacionados, respectivamente, ao valor anterior e às diferenciações de primeira ordem de valores defasados em i.

A hipótese nula é rejeitada para p-valores inferiores a 5%, ou seja, fora de um intervalo de confiança de 95%.

2.3 MODELOS ESTOCÁSTICOS

2.3.1 Suavização

A fim de mitigar a igual influência de dados antigos em relação a dados recentes, uma vez que todos dados observados são utilizados no processo de modelagem de uma série temporal, modelos de suavização surgem como alternativa por utilizarem apenas valores observados ao redor do instante *t* para estimar a tendência local.

2.3.1.1 Médias Móveis

O método para priorização de dados recentes consiste na utilização de filtros lineares, ou seja:

$$\overline{X_t} = \sum_{j=-q}^{q} a_j X_{t+j}, \quad t = q+1, \dots, T-q. \tag{2.16}$$

Onde a_j são os pesos, tal que $\sum_{j=-q}^{q} a_j = 1$, X_t é a série de entrada, $\overline{X_t}$ a série filtrada (ou suavizada) e q é o número de valores a serem considerados na operação.

Esse filtro é conhecido como *filtro de médias móveis*, e podemos classificar suas aplicações em dois grupos principais (Morettin e Toloi, 2006; Ehlers, 2009), sendo o mais relevante o método de média móvel simples, ou SMA - *Simple Moving Average,* o qual consiste na utilização de filtros lineares sem distinção nos valores dos pesos, ou seja, todos os pesos são equivalentes a um valor a tal que:

$$a = \frac{1}{2q+1} \tag{2.17}$$

Resultando na seguinte equação final:

$$\overline{X_t} = \frac{1}{2q+1} \sum_{j=-q}^{q} X_{t+j}, \quad t = n+1, \dots, N-q. \tag{2.18}$$

2.3.1.2 Suavização Exponencial Simples (SES)

A suavização exponencial visa eliminar um dos principais problemas referentes ao método SMA, que seria a não consideração de diferentes pesos dentro dos valores utilizados, ou seja, não priorizando valores mais recentes.

Matematicamente, tem-se:

$$\overline{X_t} = \sum_{k=0}^{t-1} (1-a)^k X_{t-k} + (1-a)^t \bar{X}_0, \ t = 1, \dots, T. \tag{2.19}$$

Onde $\overline{X_t}$ é o valor suavizado no instante *t*, $\bar{X}_0$ é o valor da série temporal no instante *t=1* e a é um número entre 0 e 1 denominado constante de suavização. O valor de a deve ser tal que pesos relacionados a observações antigas não sejam tão significativos quanto relacionados a observações recentes, consequentemente, valores ideias para a são valores pequenos (Morettin e Toloi, 2006).

2.3.1.3 Suavização Exponencial de Holt

O método de suavização exponencial de Holt, ou simplesmente método de Holt, consiste em um aprimoramento do método SES. Esse aprimoramento se dá através da implementação de uma nova constante de suavização, responsável pela modelagem da tendência da série, que acaba por sua vez negligenciada no método SES, gerando valores consideravelmente superiores ou inferiores dos reais (Morettin e Toloi, 2006).

As equações que descrevem o método de Holt para o modelo da série ($\overline{X_t}$) e da tendência ($\hat{T}_t$) são, respectivamente, a equação 2.20 e 2.21.

$$\overline{X_t} = aX_t + (1-a)(X_{t-1} + \hat{T}_{t-1}), \ t = 2, \dots, N. \tag{2.20}$$

$$\hat{T}_t = c(\bar{X}_t - \bar{X}_{t-1}) + (1-c)\hat{T}_{t-1}, \ t = 2, \dots, N. \tag{2.21}$$

Onde c é um novo coeficiente de suavização, entre 0 e 1. A fim de se realizar previsões, aplica-se ambas equações no domínio de tempo ($T+L$), sendo T o domínio do conjunto de observações L é o tamanho das previsões necessárias.

2.3.1.4 Método de Holt-Winters

Assim como o método anterior otimiza o método SES para séries com tendência, uma relação similar ocorre entre o mesmo e o método de Holt-Winters. O último é um aprimoramento do método de Holt para aplicações em séries com sazonalidade.

O método de Holt-Winters divide as séries temporais sazonais em duas categorias, a primeira, tendo a sazonalidade como uma componente multiplicativa e a segunda como uma componente aditiva. As equações que descrevem os casos de sazonalidade multiplicativa (2.22, 2.23) e aditiva (2.24 e 2.25) seguem abaixo:

$$\hat{S}_t = d(\bar{X}_t/\bar{X}_{t-1}) + (1-d)S_{t-s}. \quad (2.22)$$

$$\overline{X_t} = a(X_t/S_t) + (1-a)(X_{t-k} + \hat{T}_{t-1}) \quad (2.23)$$

$$\hat{S}_t = d(\bar{X}_t - \bar{X}_{t-1}) + (1-d)S_{t-s}. \quad (2.24)$$

$$\overline{X_t} = a(X_t - S_t) + (1-a)(X_{t-1} + \hat{T}_{t-1}) \quad (2.25)$$

Onde d é um terceiro coeficiente de suavização aplicado à sazonalidade e $t = s+1, \dots, T$. A equação 2.21 continua válida para descrever o modelo da tendência da série para ambos os casos.

A estimação dos coeficientes a, c e d se dá, dentre outros métodos, pelo método dos mínimos quadrados, descrito pela equação 2.26 (Morettin e Toloi, 2006).

$$LSE = min \sum_{t=1}^{T} (X_t - \hat{X}_t)^2 \quad (2.26)$$

2.3.2 Modelos Autorregressivos

Regularmente, a hipótese de que uma série temporal é representada por duas componentes, determinística e estocástica, não é suficiente para obtenção de um modelo condizente. Nesses casos, modelos de suavização não fornecem previsões adequadas, e utiliza-se então a classe de modelos autoregressivos.

Modelos autoregressivos, ou AR (*autoregressive*), utilizam valores defasados da variável observada, visando capturar a autocorrelação da série. Matematicamente, pode-se então equacionar a variável observada de acordo com a equação 2.27.

$$\hat{X}_t = \sum_{i=1}^{p} \emptyset_i X_{t-i} + \varepsilon_t \quad (2.27)$$

Onde $\emptyset$ é o coeficiente e p é a ordem de autoregressão, respectivamente (Pal e Prakash, 2017). Aplicando conceito de operador de defasagem (*B*) tal que $B^i X_t = X_t - X_{t-i}$, temos as seguintes equações para o modelo autoregressivo de ordem *p,* denotado AR(p):

$$\emptyset(B) = 1 - \emptyset_1 B - \emptyset_2 B^2 - \cdots - \emptyset_p B^p \quad (2.28)$$

$$\emptyset(B)\hat{X}_t = \varepsilon_t \quad (2.29)$$

Sendo $\hat{X}_t = X_t - \mu$ (Box et al., 2015).

Ainda, modelos autoregressivos podem ser combinados com outras ferramentas de modelagem de séries temporais já mencionadas neste trabalho, como médias móveis e diferenciação (Pal e Prakash, 2017)

2.3.2.1 ARMA

Um modelo ARMA (*Autorregressive-moving-average*) consiste na aplicação mista de um modelo autoregressivos e de médias móveis. Os mesmos, de acordo com (Morettin e Toloi, 2016), são extremamente comuns em séries econômicas, e fornecem modelos em muitos casos satisfatórios com uma quantidade relativamente pequena de parâmetros. Modelos ARMA são classificados quanto à ordem de autoregressão (*p*) e a quantidade de valores anteriores considerados pelo método de médias móveis (*q*) (Tamura, 2013).

Um modelo de médias móveis (MA) pode ser descrito em função do ruído da série, como demonstram as equações 2.30 e 2.31:

$$\theta(B) = 1 - \theta_1 B - \theta_2 B^2 - \cdots - \theta_q B^q \quad (2.30)$$

$$\hat{X}_t = \theta(B)\varepsilon_t \quad (2.31)$$

Onde θ_i são os coeficientes de média móvel. Consequentemente, define-se um modelo ARMA(*p*,*q*) através da seguinte equação:

$$\emptyset(B)\hat{X}_t = \theta(B)\varepsilon_t \quad (2.32)$$

2.3.2.2 ARIMA

De acordo com Tamura (2013), modelos AR, MA e ARMA são de modo geral suficientes para descrição e previsão de séries temporais estacionárias, ou seja, sérias cujas características obedecem às condições descritas na seção 2.2.1. Para modelagem e previsão de séries com tendência, utiliza-se o modelo ARIMA (*Autoregressive-integrated-moving-average*).

O modelo ARIMA consiste no aprimoramento do modelo ARMA através de *d* diferenciações da série temporal com o intuito de se remover a tendência e tornar a série estacionária. A equação 2.33 descreve o modelo ARIMA.

$$\emptyset(B)(1 - B)^d \hat{X}_t = \theta(B)\varepsilon_t \quad (2.33)$$

Um modelo ARIMA é denotado por ARIMA(p,d,q).

2.3.2.3 SARIMA

Visando englobar séries sazonais, o modelo ARIMA é generalizado, e obtém-se, então, o modelo SARIMA (*Seasonal-autoregressive-integrated-moving-averages*), que adiciona três parâmetros, P, D e Q, referentes à, respectivamente, autoregressão, diferenciação e média móvel da componente sazonal. Logo, as equações referentes ao modelo SARIMA com sazonalidade multiplicativa são:

$$\Theta(B^s) = 1 - \Theta_1 B - \Theta_2 B^{2s} - \cdots - \Theta_Q B^{Qs} \quad (2.34)$$

$$\Phi(B^s) = 1 - \Phi_1 B - \Phi_2 B^{2s} - \cdots - \Phi_P B^{Ps} \quad (2.35)$$

$$\Phi(B^s)\emptyset(B)(1 - B)^d(1 - B^s)^D \hat{X}_t = \Theta(B^s)\theta(B)\varepsilon_t \quad (2.36)$$

Onde *s* é o período sazonal e Φ e Θ são os coeficientes de autoregressão e média móvel, respectivamente (Ehlers, 2009).

Um modelo SARIMA é, analogamente a um modelo ARIMA, denotado como SARIMA(p,d,q)(P,D,Q,s).

2.3.2.4 Critérios de informação

É comum em modelagem de dados reais o aumento do número de parâmetros a fim de se aumentar a acuracidade do modelo. Esse aumento, quando ocorre de maneira exagerada, pode comprometer futuras previsões, por perder sua robustez. Esse fenômeno é denominado *overfitting* (excesso de ajuste). Com o intuito de se estimar os parâmetros *p*, *d* e *q*, além de *P*, *D* e *Q* em caso de série sazonal, procurando se evitar a adição exagerada de variáveis aos modelos, faz-se uso de critérios que além de levarem em conta a qualidade do ajuste penalizam o número de parâmetros.

Um critério de informação prioriza o aumento da acuracidade do mesmo, através da variância residual, porém penaliza adição de parâmetros. Os critérios mais consagrados no estudo de séries temporais são o Critério de Informação de Akaike, ou AIC (***Akaike's** Information Criterion*) e o Critério de Informação Bayesiano, BIC (*Bayesian Information Criterion*), descritos pelas equações 2.37 e 2.38, respectivamente.

$$AIC = N + T\log(\hat{\sigma}^2) \qquad (2.37)$$

$$BIC = N\log T + -2\log(\hat{\sigma}^2) \qquad (2.38)$$

Onde *N* é o número de parâmetros e $\hat{\sigma}^2$ é a variância residual do modelo, análoga à equação 2.6 (Ehlers, 2009).

2.4 VARIÁVEIS EXÓGENAS

A modelagem de séries temporais consiste na análise de sistemas dinâmicos, envolvendo suas variáveis de entrada e de saída. Maçaira et al. (2018) divide as séries temporais em dois grupos, univariadas e multivariadas, sendo o primeiro composto por séries descritas pelos modelos estudados na seção 2.3. O segundo grupo, entretanto, abrange séries temporais que, além de serem funções de suas componentes defasadas e do tempo, são diretamente relacionadas a variáveis externas, as mesmas definidas como variáveis exógenas ou explicativas. Ainda de acordo os autores, exemplos de aplicações de variáveis exógenas são a modelagem da concentração de SO_2 utilizando dados relacionados a fatores meteorológicos, como vento, temperatura e umidade, previsão de elasticidade de preço em demanda residencial de água utilizando localização, entre outros.

Há diversos métodos desenvolvidos na literatura para inserção de variáveis exógenas na análise e modelagem de séries temporais, sendo os mais consagrados o método de regressão, redes neurais artificiais e ARIMAX, respectivamente (Maçaira et al., 2018). Métodos regressivos e ARIMAX são abordados nesta seção, enquanto redes neurais artificiais são abordadas mais detalhadamente na seção 2.5.

2.4.1 ARIMAX

Modelos ARIMAX (ou SARIMAX) é um modelo ARIMA (ou SARIMA) com adição de uma ou mais variáveis exógenas. Evidenciando $\hat{X}_t$ na equação 2.36, e adicionando os termos relacionados à variável exógena *y*, temos a equação 2.39 que descreve um modelo SARIMAX.

$$\hat{X}_t = f(y) + \left(\frac{\Theta(B^s)\theta(B)\varepsilon_t}{\Phi(B)\emptyset(B)(1-B)^d(1-B^s)^D}\right) \qquad (2.39)$$

A função $f(y)$ pode apresentar diversas formas, como um modelo de regressão ou ainda uma função de variáveis *dummies* (1 ou 0) (Suhartono et al., 2015).

2.4.2 Regressão

Método mais aplicado em análises de séries temporais multivariadas, métodos regressivos assumem que a relação entre a variável observada e a variável exógena não

variam ao longo do tempo. Se uma variável observada depende unicamente de uma variável exógena *y* e de componentes estocásticas, ela pode ser definida pela equação 2.40.

$$X_t = \sum_{i=1}^{n} \beta_i y_t^i + \varepsilon_t \qquad (2.40)$$

Sendo β_i os coeficientes de um polinômio de regressão de ordem *n* (Nelder e Wedderburn, 1972).

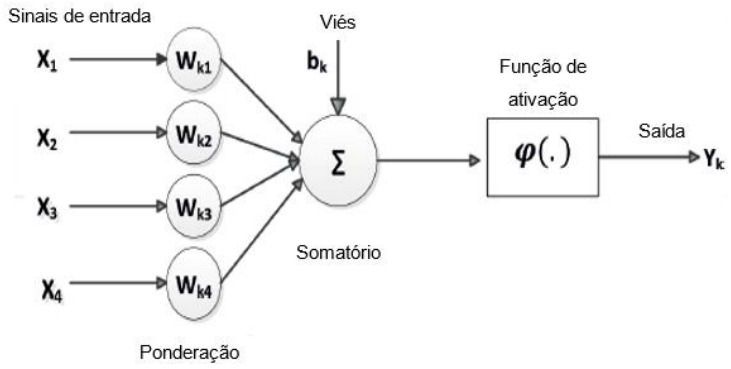
Para previsões, modelos exógenos necessitam dos valores futuros das variáveis exógenas, o que pode dificultar o uso dessa ferramenta em casos onde a variável exógena não é conhecida em todo domínio (*e.g.* feriados em séries semanais).

2.5 REDES NEURAIS ARTIFICIAS

Ainda que os métodos descritos nas seções anteriores consigam abordar grande parte da problemática envolvida em muitas séries temporais, há ainda uma gama de casos em que os parâmetros envolvidos nos mesmos não serão suficientes para uma modelagem acurada do comportamento da variável observada ao longo do tempo. Isso se deve à dificuldade (ou ainda impossibilidade) de se identificar variáveis exógenas para o sistema, ou ainda de se definir essa relação, quando constatada.

Redes neurais artificiais (RNA) são estruturas matemáticas que apresentam uma certa analogia com a rede de neurônios de um cérebro humano, de tal modo que cada unidade da rede é chamada de neurônio. O conjunto de dados atribuído a uma rede neural é treinado um determinado número de vezes (épocas) e recebe, para cada dado, um peso (Nelson, 2017; Baffa et al., 2008). A Figura 2.3 ilustra um modelo de neurônio artificial.

Figura 2.3 - Modelo de neurônio artificial de McCulloch e Pitts.



Fonte: Guesmi et al. (2019).

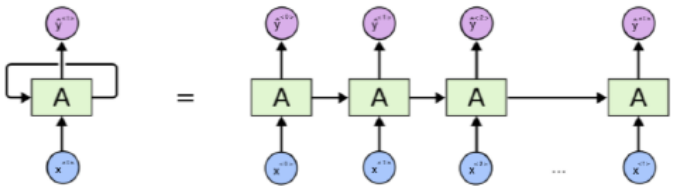
Uma rede neural que obedece à esquemática mostrada na Figura 2.3, ou seja, uma função direta de suas entradas, com fluxo unidirecional de informação, é denominada rede neural *feed-forward*. Por possuir uma arquitetura tal que a informação passa apenas uma vez por cada unidade, não permitindo a mesma armazenar memórias de dados de entrada prévios, redes neurais *feed-forward* não são adequadas para o estudo de dados sequenciais (*e.g.* séries temporais) (Castelão, 2018).

2.5.1 Redes neurais recorrentes

Redes neurais recorrentes (RNN – *Recurrent Neural Networks*) possuem uma arquitetura (ilustrada pela Figura 2.4) que torna possível a geração de um valor de saída que leve também em conta informações do passo anterior. São, portanto, capazes de fornecer modelos de complexidade muito superior que, apesar de aumentarem a dificuldade de aprendizado,

conseguem de fato aumentar a gama de problemas possíveis de se resolver com técnicas de *Machine Learning*.

Figura 2.4 - Estruturas de uma rede neural recorrente simplificada (esquerda) e desenrolada (direita).



Fonte: Castelão (2018).

Entretanto, de acordo com Castelão (2018), há casos onde informações muito antigas ainda são relevantes, e algumas estruturas de RNN ainda são insuficientes para aplicação nos mesmos. Isso fica evidente em casos de séries temporais sazonais, ou até mesmo com alta ordem de autoregressão; para tal, utiliza-se uma arquitetura específica de rede neural recorrente, conhecida como rede neural de curto e longo prazo, ou LSTM (*Long short-term memory*).

2.5.2 Redes neurais LSTM

Redes neurais LSTM surgiram para aumentar a capacidade de memória de uma rede neural. Isso se dá através da utilização de novas unidades, chamadas portões, as quais mantêm o fluxo de erro constante (Nelson, 2017).

Chen (2017) divide em três as etapas de confecção da arquitetura de uma rede neural LSTM:

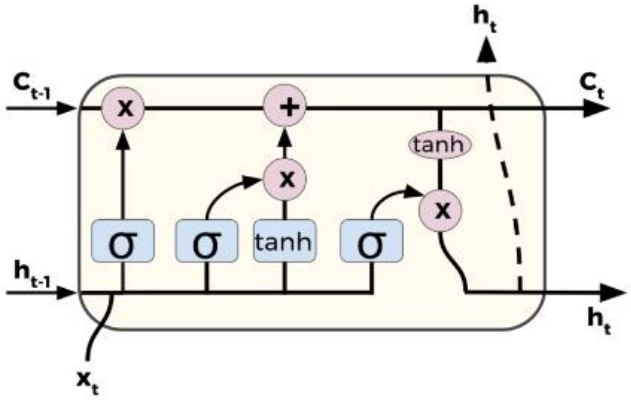
1) Utilização de um mecanismo de esquecimento: mecanismo que interpreta se um dado de entrada deve ser de fato utilizado ou descartado.

2) Utilização de um mecanismo de salvamento: mecanismo responsável por salvar as partes úteis dos dados de entrada na memória a longo prazo.

3) Transformar memória de longo prazo em memória útil: o modelo deve aprender quais partes da memória a longo prazo devem ser utilizados a cada dado instante.

A Figura 2.5 ilustra esses mecanismos em um neurônio de uma rede neural LSTM:

Figura 2.5 - Estruturas de um neurônio de uma rede neural recorrente LSTM.



Fonte: Portilla (2019).

Onde C_t representa a memória de longo prazo (ou estado) do modelo, x_t são os dados de entrada e h_t os dados de saída no instante t. Ainda, no modelo LSTM, a função sigmoide (σ) retorna valores de 0 a 1, e atua como um filtro que define a passagem ou não do sinal. A função tangente hiperbólica (*tanh*) atua na criação de novos candidatos para a memória de longo prazo, variando-os de -1 a 1. Ambas as definições para este caso seguem descritas pelas equações 2.41 e 2.42 (Castelão, 2018).

$$\sigma(x) = \frac{1}{1+e^{-x}} \quad (2.41)$$

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \quad (2.42)$$

Castelão (2018) e Portilla (2019) descrevem as funções de cada um dos portões em uma rede neural LSTM:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \quad (2.43)$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \quad (2.44)$$

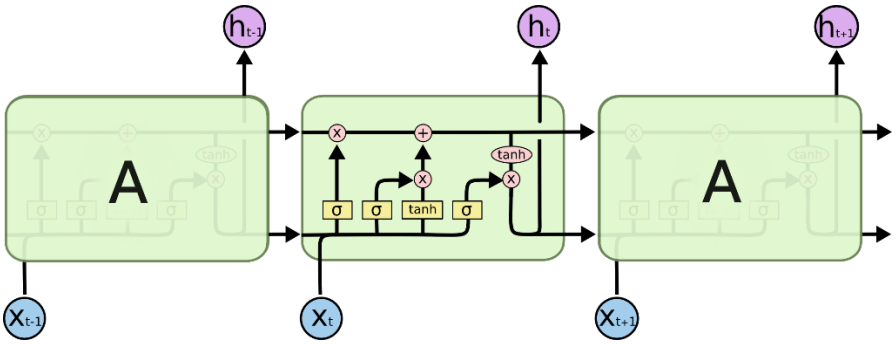
$$C_t = f_t C_{t-1} + i_t(tanh(W_C[h_{t-1}, x_t] + b_C)) \quad (2.45)$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad (2.46)$$

$$h_t = o_t tanh(C_t) \quad (2.47)$$

Onde f_t é o portão de esquecimento (*forget gate*), i_t é o portão de entrada (*input gate*) e o_t é o portão de saída (*output gate*). C_t é a memória a longo prazo, ou estado (*cell state*). Os parâmetros de ajuste W e b são, respectivamente os pesos e os ajustes por viés que ocorrem dentro da célula da rede neural. Replicando essa estrutura para todas unidades da rede neural, temos a arquitetura de uma rede neural recorrente LSTM, ilustrada pela Figura 2.6.

Figura 2.6 – Arquitetura de uma rede neural recorrente LSTM.



Fonte: Olah (2015).

Devido essa complexa estrutura, na qual a informação flui entre diversas camadas dentro da célula, a rede neural LSTM se destaca em sistemas dinâmicas e dados sequenciais, e será a ferramenta utilizada neste trabalho.

3 REVISÃO BIBLIOGRÁFICA

O entendimento de sistemas e seu comportamento ao longo do tempo é fundamental em diversas áreas de estudo, bem como, através desse entendimento, a capacidade de se gerar previsões futuras para esse sistema. Dito isso, diversos estudos têm sido realizados se fazendo uso de ferramentas distintas, das mais simples, como modelos de suavização, a mais sofisticadas, como redes neurais. Esta seção tem como objetivo estudar a aplicação dessas técnicas para previsão de séries temporais em diversos campos, bem como os mais relevantes para os casos estudados neste trabalho.

Uma das áreas mais proeminentes, responsável por grande contribuição no estudo de séries temporais, é a área das ciências econômicas, na qual estudos de séries temporais hoje são indispensáveis (Nesser, 2016). Tamura (2013) trabalhou com previsões de preços de ações através da utilização de modelos AR com regressão linear em variável exógena (um índice diário de ações), estudando a ordem da utilização dos dois métodos. O autor concluiu que a ordem das regressões utilizadas infere na adequação do modelo, obtendo um modelo de regressão mista (autoregressão e regressão linear exógena simultaneamente) como o mais apropriado para o seu caso.

Por se tratar de uma área muito dinâmica onde a identificação de variáveis exógenas é de extrema dificuldade, o que torna inviável a utilização de modelos autoregressivos, pesquisadores da área econômica, especialmente focados no mercado de ações, costumam empregar técnicas mais sofisticadas na modelagem de séries temporais. Nelson (2017) e Castelão (2018) utilizaram redes neurais artificiais LSTM para previsões de ações e ambos conseguiram atingir resultados razoáveis. Enquanto o primeiro obteve sucesso em prever comportamento de crescimento ou queda em uma ação, Castelão (2018) obteve previsões adequadas para 50 dias. Dentre outras pesquisas na área, destaca-se o método desenvolvido por Cao et al. (2019), o qual combina redes neurais LSTM com métodos de decomposição, obtendo previsões otimizadas na área de séries financeiras.

Outra área em que aplicações de ferramentas de análise de séries temporais são relevantes é a área de PCP (Planejamento e Controle de Produção). Lima et al. (2015) utilizaram o método de Holt-Winters, comparando as suas diferentes interpretações para a componente sazonal – aditiva e multiplicativa –, para a demanda de aço de uma empresa de corte e dobra de aço das regiões Norte e Nordeste do Brasil em seis meses. Os autores se baseiam no trabalho de Lustosa et. al (2008), que classifica modelos de suavização e autorregressivos como adequados para modelagem e previsão de séries temporais de demanda. Os resultados obtidos foram satisfatórios para Holt-Winters com sazonalidade aditiva.

Estreitando a análise para a aplicação de métodos autoregressivos na área de PCP, temos os estudos de Xavier (2016) acerca da produção mensal de leite de vaca na província de Victoria, Argentina, bem como a previsão de vendas de motocicletas estudada por Walter et al. (2013). Ambos estudos envolveram o método SARIMA, e ainda concluíram que a aplicação do método é viável para ambos os casos. Xavier (2016) ainda argumentou que modelos SARIMA superam modelos paramétricos (modelagem fora do domínio do tempo) em casos de sazonalidade bem definida.

Taşpinar et al. (2013) estudou predições na área de demanda de combustíveis fósseis, um dos focos deste trabalho, ao analisar a eficiência de modelos SARIMAX em prever valores de consumo diário de gás natural na Turquia, utilizando variáveis climáticas como exógenas. O autor constata que o modelo para este caso foi superior em relação a aplicação de redes neurais; entretanto, o método de redes neurais aplicado não foi o de redes neurais recorrentes LSTM. Séries temporais de demanda energética são de complexas modelagem e previsão por não possuir uma tendência bem definida (Mousavi e Ghavidel, 2019), devido a influência de diversos fatores externos. Ainda, em um estudo recente, Sagheer e Kobt (2019) constataram a alta aplicabilidade de redes neurais DLSTM (*Deep* LSTM), um aprimoramento

do método LSTM, para séries temporais envolvendo combustíveis fósseis, visto que, ao modelarem uma série de produção de petróleo, obtiveram ajustes muito superiores aos obtidos por métodos tradicionais.

Além de amplamente aplicados por pesquisadores de áreas sob forte influência de fatores econômicos, métodos de análise e previsão de séries temporais possuem constante aplicação em estudos de fenômenos físicos e naturais. Dentre outros pesquisadores desse campo de estudo, Tibulo (2014) realizou um estudo detalhado acerca de previsões de umidade relativa do ar, comparando modelos estudados neste trabalho, constatando a viabilidade da aplicação de modelos da classe ARMAX para esse fenômeno. Ainda, tratando-se de fenômenos naturais, Ozoegwu (2019) constatou que aplicação de redes neurais para a previsão de médias mensais de radiação solar em determinadas regiões pode ser aplicada com sucesso; ainda, as previsões possuem uma acuracidade tal que podem ser utilizadas em processos de planejamento agrícola.

À medida que métodos cada vez mais complexos são desenvolvidos em diversos campos de estudo, e constatando a aplicação viável em processos físicos, é possível deduzir que ferramentas de estudo de séries temporais possuem potencial de aplicação em áreas de engenharia. Comprovando essa dedução, diversos estudos têm sido conduzidos nas mais diversas áreas da engenharia fazendo uso de tais ferramentas. Dentre eles, Miranda et al. (2009) teve sucesso em elaborar previsões semanais para carga elétrica com dados de alta frequência utilizando o método de Holt-Winters com adição de feriados como variável exógena.

Dentro do campo da engenharia, destacam-se potenciais aplicações para a metodologia estudada neste trabalho: prevenção de acidentes e análise de tempo de degradação de equipamentos. Modelos ARIMA têm sido implementados, combinados com outras ferramentas, para determinar a vida útil de motores de aeronaves (Ordóñez et al., 2019) e prever o estado de um trocador de calor em instalações geotérmicas (Baruque et al., 2019). No primeiro caso, a aplicação do modelo ARIMA é uma etapa inicial de um complexo algoritmo; não obstante, resultados preditivos encorajadores comparados com outros métodos são alcançados. O último obteve previsões anuais de temperatura média satisfatórias, com erros inferiores a 1 °C.

Conclui-se por esta revisão bibliográfica que, nas mais diversas áreas de estudo, modelos autoregressivos costumam fornecer resultados satisfatórios, mesmo que em alguns casos de maior complexidade sejam acompanhados de outras ferramentas. Ainda, conclui-se que, apesar de estudos na área de PCP serem já uma realidade, estudos relacionados à previsão de produção e demanda de petróleo e combustíveis derivados são muito recentes e ainda escassos na literatura.

4 METODOLOGIA

Esta seção tem como objetivo descrever a metodologia elaborada para a execução deste trabalho, bem como o embasamento de cada passo, quando necessário. A metodologia difere para cada caso estudado, o que será aprofundado na seção 5. Todo trabalho será realizado com o uso da linguagem de programação Python, com o uso dos seguintes pacotes: *Pandas, Numpy, Statsmodels, Pmdarima, Keras* e *Tensorflow.*

4.1 VISUALIZAÇÃO DA SÉRIE TEMPORAL

O primeiro passo para escolher a melhor classe de modelos para uma série temporal é através da identificação das suas principais componentes, ou seja, como se comporta sua tendência e se há ou não presença de componente sazonal.

A detecção da sazonalidade se dá através da análise dos gráficos ACF e PACF. A mesma análise será realizada para detecção de autoregressão.

4.2 DETECÇÃO DE VARIÁVEL EXÓGENA

As variáveis exógenas a serem analisadas serão definidas por suposição. A comprovação ou não de existência de dependência da variável exógena por parte da variável exógena será avaliada através de gráficos de correlação.

4.3 SEPARAÇÃO DA SÉRIE TEMPORAL

O estudo de previsões de séries temporais consiste na divisão da mesma em dois conjuntos: “treino”, o qual é composto pelos dados iniciais da série temporal, e “teste”, consistindo nos dados finais da mesma. Os ajustes são obtidos através da modelagem da série de treino, a partir dos quais são realizadas previsões, que são então comparadas com o conjunto de teste. Consequentemente, o conjunto de teste deve possuir o mesmo tamanho da previsão desejada. As seguintes equações descrevem precisamente essa divisão:

$$Tr_t = (X_1, X_2, \dots, X_{T-fc}) \qquad (4.1)$$

$$Ts_t = (X_{T-fc+1}, X_{T-fc+2}, \dots, X_T) \qquad (4.2)$$

Onde Tr_t e Ts_t são, respectivamente, o conjunto de treino e teste, fc é o tamanho das previsões e T é o tamanho do conjunto de dados. Para este trabalho, serão realizadas previsões 12 meses para o estudo de fornecimento de gasolina A Rio Grande do Sul e uma separação de treino e teste em 70% e 30%, respectivamente, do tamanho da série para os estudos da unidade de desidratação de gás natural por TSA.

4.4 SUAVIZAÇÃO

Para a escolha do método de suavização a ser aplicado, será levado em conta o resultado da análise anterior. Caso não seja possível descartar a presença de sazonalidade, será utilizado o método de Holt-Winters.

4.5 AUTOREGRESSÃO

O critério de escolha dos parâmetros do modelo autoregressivos – ARIMA, SARIMA, ARIMAX ou SARIMAX, dependendo dos resultados das análises de sazonalidade e variável exógena – a ser utilizado se dará através da aplicação do critério de informação AIC, descrito anteriormente no item 2.3.2.4 deste trabalho.

4.6 REDES NEURAIS LSTM

A fim de escolher a arquitetura ideal de uma rede neural LSTM, deve-se levar em conta 3 fatores principais: o número de valores passados utilizados para realizar a predição de um ponto (*window*), o número de interações, ou épocas (*epochs*), entre a rede neural e o conjunto de treino e a quantidade de neurônios. Para cada caso estudado, serão estudados diversos valores diferentes para o número de células utilizadas, visto que não há um critério de estimação consagrado para tal. Dito isso, é esperado que, tendo uma maior complexidade em uma série temporal o número de células a serem utilizadas será maior do que em casos de séries mais simples.

O número de épocas tem como objetivo "familiarizar" a rede neural com o conjunto de dados. Para cada interação, a rede neural estima pesos e obtém um modelo. Esse modelo é então avaliado com um critério; no caso deste trabalho, será através do EQM (Erro Quadrado Médio), dado por:

$$EQM = \sum_{t=1}^{n} (\widehat{Tr}_t - Tr_t)^2 \qquad (4.3)$$

Onde $\widehat{Tr}_t$ representa os valores obtidos pelo método para cada ponto equivalente do conjunto de treino. À medida em que se aumenta o número de épocas, diminui-se o valor de *EQM*. Ainda, deve-se evitar a adição excessiva de épocas, para não haver *overfitting*, e consequentemente o comprometimento das previsões. Analogamente aos modelos ARIMA, a notação utilizada neste trabalho para redes neurais LSTM será LSTM(*Ne,w,ep*), sendo *Ne, w* e *ep* o número de neurônios, *window* e o número de épocas, respectivamente.

A minimização do erro quadrado médio se dá através do ajuste dos pesos da rede neural a partir de pesos iniciais aleatórios fazendo-se uso de um otimizador. No caso deste trabalho, o otimizador e o processo de inicialização aleatória utilizados foram, respectivamente, o otimizador Adam (*Adaptive Moment Estimation*) e o processo de inicialização de Glorot, ferramentas cujos conceitos não serão abordados neste trabalho (Kingma e Ba, 2015; Glorot e Bengio, 2010).

Ainda, essa metodologia é facilitada utilizando-se dados de menor amplitude; para isso, aplica-se uma normalização no conjunto de dados entre 0 e 1, conforme a equação 4.4.

$$X_t^s = \frac{X_t - Min(X_t)}{Max(X_t) - Min(X_t)} \qquad (4.4)$$

Onde X_t^s é a série normalizada e $Max(X_t)$ e $Min(X_t)$ são, respectivamente, o maior e o menor valor da série. Assim, todos os pontos da série são ajustados para valores entre 0 e 1, sendo 0 o menor, e 1 o maior (Pal e Prakash, 2017).

4.7 AVALIAÇÃO DAS PREVISÕES

As previsões serão avaliadas através de dois critérios:

- Estacionaridade dos resíduos.
- REQM (Raiz do Erro Quadrado Médio).

A REQM nada mais é que $\sqrt{EQM}$. Sua aplicação é útil para maior compreensão da magnitude do erro, especialmente para avaliação de erro em viés, ou seja, se, em média, a previsão se aproxima da série real, por ter a mesma unidade da variável observada. A estacionaridade dos resíduos será avaliada pelo Teste Aumentado de Dickey-Fuller (ADF).

4.8 ANÁLISE DO COMPORTAMENTO DINÂMICO

Para se avaliar a robustez dos modelos, será repetido todo o procedimento descrito nesta seção, alterando-se o ponto de previsão; assim, obtendo previsões realizadas em pontos anteriores e comparando os resultados com previsões mais recentes.

5 ESTUDO DE CASO

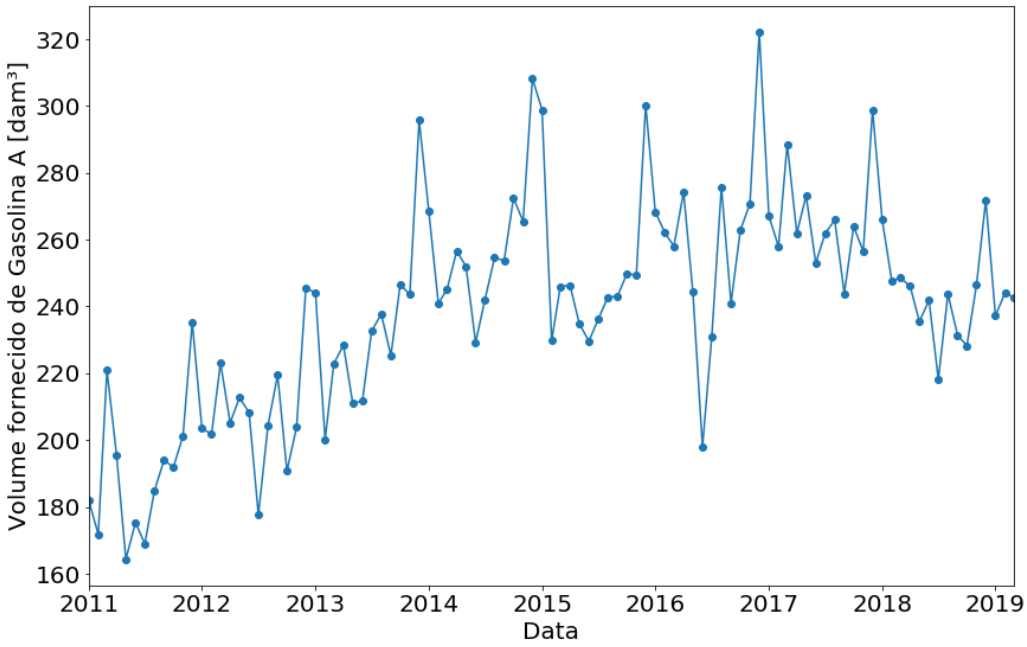
5.1 FORNECIMENTO DE GASOLINA A NO RIO GRANDE DO SUL

Em 2018, o Brasil era o sétimo maior consumidor de produtos derivados de petróleo do mundo, tendo ainda um consumo *per capita* muito inferior aos demais países do ranking; ou seja, com um grande potencial de crescimento (ANP, 2018). Gasolina, particularmente, é o combustível derivado de petróleo cujo consumo apresentou maior crescimento de 2016 a 2017.

Dito isso, é evidente que a previsão de comportamentos futuros no mercado de Gasolina A é fundamental para o PCP (Planejamento e Controle da Produção) de refinarias e distribuidoras. Este trabalho irá aplicar a metodologia apresentada no capítulo anterior à série temporal de fornecimento de Gasolina A (componente principal da gasolina comum, compondo 73% da mesma) no estado do Rio Grande do Sul. Os dados são mensais, de janeiro de 2011 a março de 2019, e estão disponíveis no site da ANP.

Aplicando a metodologia, o primeiro o passo é a visualização da série. A unidade original dos dados da ANP é metro cúbico (m³), porém, para melhor aplicação dos métodos, os valores foram convertidos para decâmetro cúbico (dam³) através de uma divisão por 1000. A série será chamada de “VOL”, e é ilustrada pela Figura 5.1.

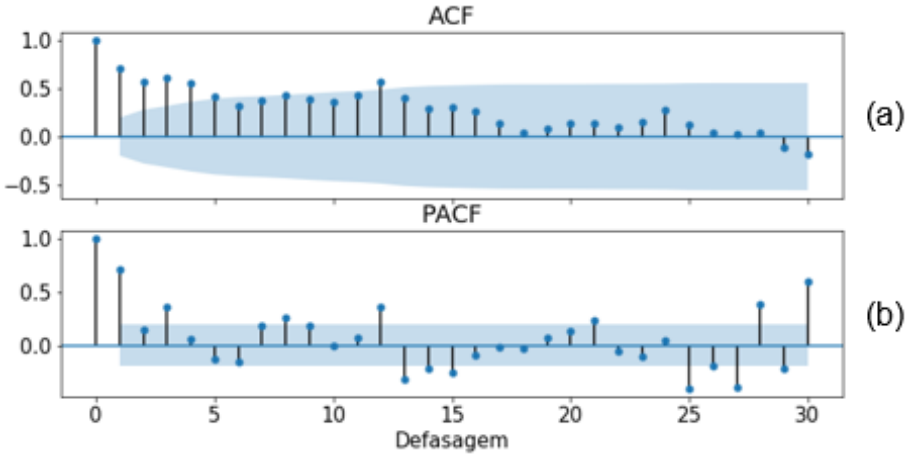
Figura 5.1 – Fornecimento de Gasolina A no estado do Rio Grande do Sul.



Fonte: ANP (2019).

Através da visualização da série, vemos que a série apresenta variações de tendência notáveis, conforme o esperado pelo estudo da literatura. A visualização também indica presença de sazonalidade, que pode ser comprovada pelos gráficos de ACF e PACF (Figura 5.2).

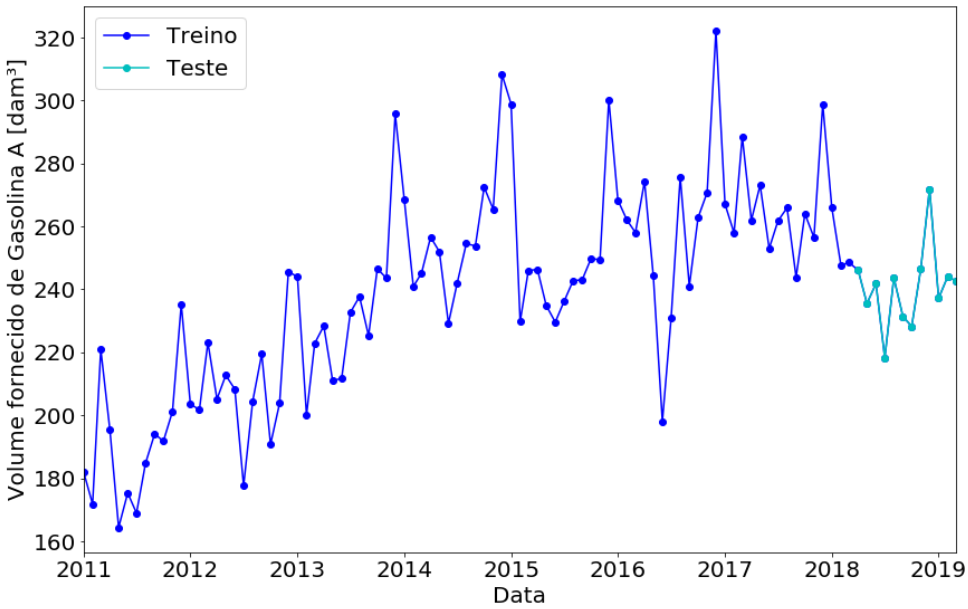
Figura 5.2 – Gráficos de autocorrelação **(a)** e autocorrelação parcial **(b)** da série VOL (30 primeiras defasagens).



Fonte: Elaborado pelo autor (2019).

O gráfico de ACF apresenta picos de autocorrelação a cada 12 defasagens, logo, confirma a presença de sazonalidade com período de 12 meses, o que é esperado em uma série de demanda com dados mensais. Em seguida, é necessário realizar a divisão treino-teste da série. As previsões estudadas para a série VOL terão comprimento de 12 meses. A divisão é descrita pela Figura 5.3.

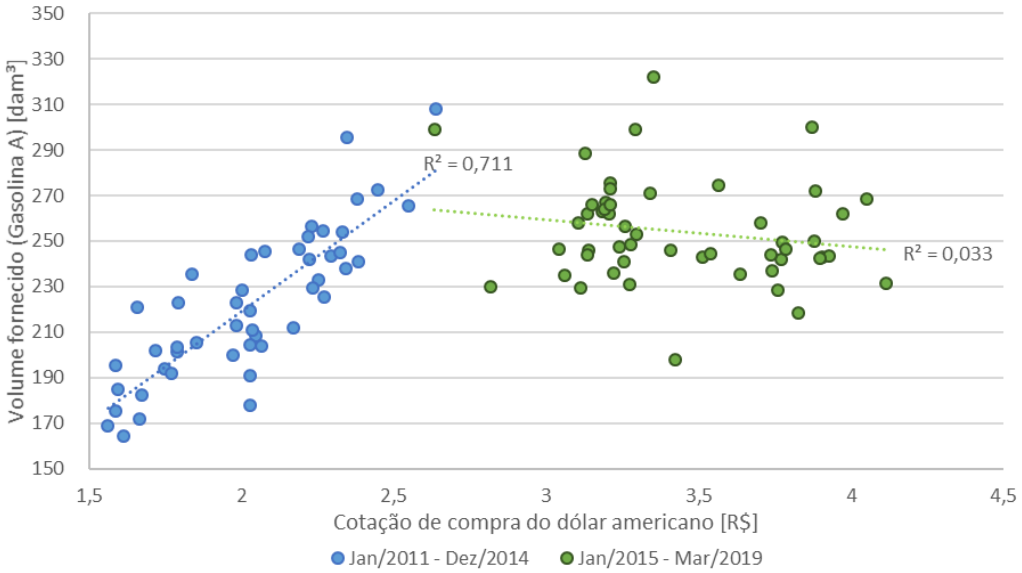
Figura 5.3 – Divisão Treino-teste para previsões de 12 meses.



Fonte: Elaborado pelo autor (2019).

Após as constatações realizadas pelos procedimentos anteriores, os modelos estocásticos escolhidos para modelagem e previsão são SARIMA e Holt-Winters, pela presença de sazonalidade. O próximo passo é a verificação de possível interferência de variáveis exógenas. Por ser um mercado extremamente dependente do comércio exterior, a variável exógena escolhida para análise é a cotação do dólar americano. O gráfico de correlação entre as duas variáveis é demonstrado pela Figura 5.4.

Figura 5.4 – Correlação entre a série VOL e a cotação de compra do dólar americano.



Fonte: Elaborado pelo autor (2019).

Observando a Figura 5.4, pode-se observar duas zonas distintas de correlação, a primeira, na esquerda do gráfico, apresenta presença de correlação, enquanto a segunda, na direita, mostra ausência total de correlação. Visto que a cotação do dólar apenas aumentou a longo prazo desde 2011, é possível concluir que a correlação forte entre a série VOL e a cotação dólar ocorreu durante um ciclo, deixando de existir perante uma mudança de comportamento brusca da série. Analisando o conjunto de dados, constata-se que este ciclo encerra em dezembro de 2014.

Os motivos desta variação podem ser diversos, como instabilidade política ou crises globais de petróleo; entretanto, não faz parte do escopo deste trabalho defini-las. O que se pode concluir é que, para previsões atuais, não será possível o aprimoramento das previsões realizadas pelos modelos estocásticos utilizando a cotação do dólar como variável exógena. Porém, analisando-se o comportamento dinâmico do modelo, a variável exógena pode ser utilizada para períodos anteriores a janeiro de 2015.

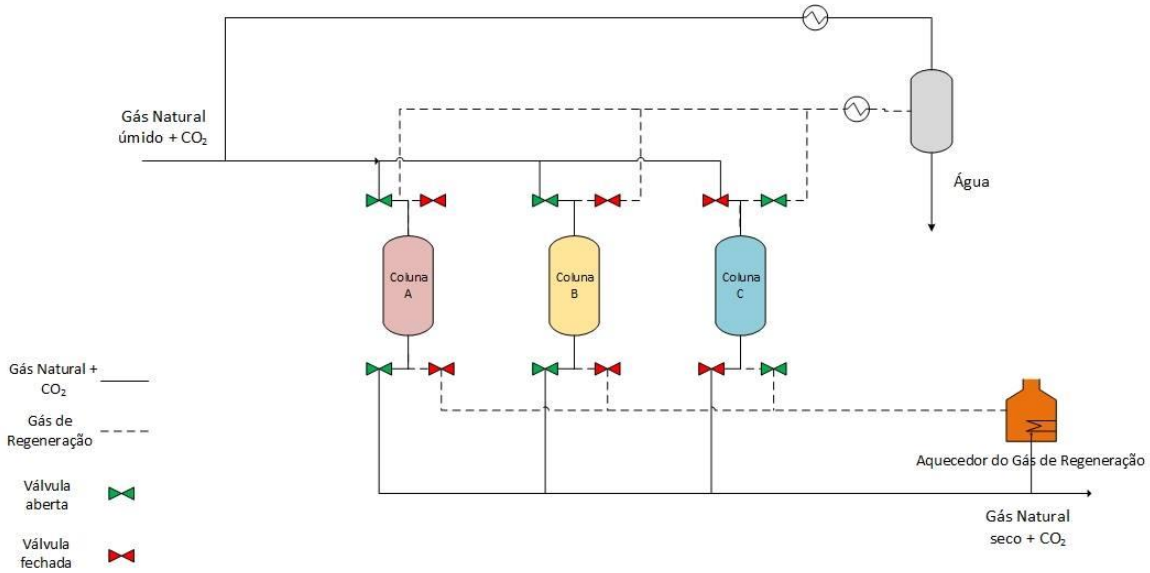
5.2 UNIDADE TSA PARA DESIDRATAÇÃO DE GÁS NATURAL

Uma das principais impurezas da extração de gás natural, a quantidade excessiva de água, quando não devidamente removida, pode acarretar diversas complicações na linha de produção, como corrosão e incrustação de hidratos em tubulações de poço (De Marco, 2019). Ainda, a temperatura de orvalho de água do gás natural é um parâmetro especificado no produto final pela Resolução ANP Nº16 (ANP, 2019b). Dito isso, é de vital importância que processos de desidratação de gás natural sejam eficientes e controlados.

Um dos processos mais aplicados para desidratação do gás natural é o TSA (*Temperature Swing Adsorption*), que consiste na adsorção a temperatura constante e dessorção do contaminado (regeneração do leito) através da alteração de temperatura. O processo TSA é amplamente utilizado quando o composto de remoção desejada é fortemente adsorvido ou quando o processo de dessorção ocorre a pequenas variações de temperatura (Fonseca, 2011; De Marco, 2019).

A unidade estudada possui três colunas de adsorção, a qual é realizada através do uso de peneiras moleculares. A regeneração do leito se dá pela passagem de gás natural seco aquecido, reciclado da corrente de produto final. A mesma é descrita pela Figura 5.5 (De Marco, 2019).

Figura 5.5 – Unidade TSA de desidratação de gás natural.



Fonte: De Marco (2019).

Conforme visto na Revisão Bibliográfica, análises de séries temporais possuem alta aplicabilidade na área de predição de degradação de equipamentos e controle de processos. Uma variável que pode ser utilizada como parâmetro para controlar a vida útil de uma coluna de adsorção é o fator de resistência ao escoamento (*R*).

$$R = \sqrt{\Delta P / \rho}\Big/_{F} \qquad (5.1)$$

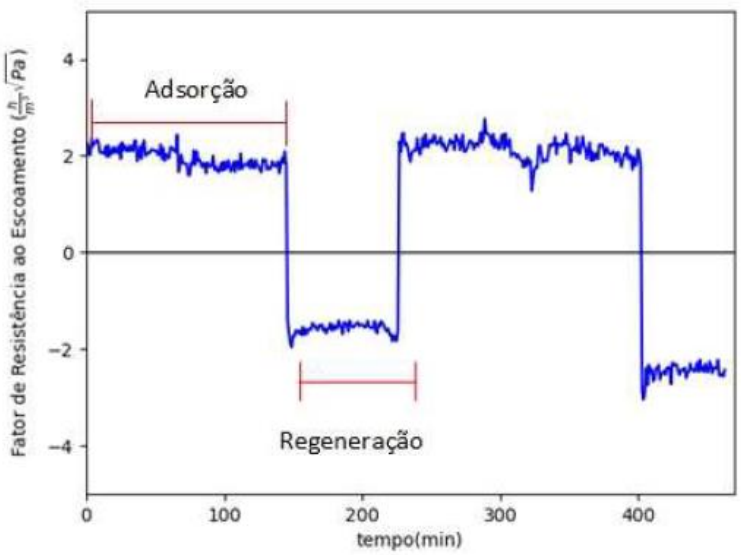
Onde ΔP, ρ e F são a diferença de pressão na coluna, a densidade do gás a vazão na coluna, respectivamente (De Marco, 2019).

Estimando-se corretamente valores futuros de *R,* não só é possível se antecipar o fim da vida útil da coluna de adsorção, como também otimizar o custo-benefício do processo, aumentando o seu tempo de utilização caso esteja abaixo da sua capacidade. Favarini (2018) realizou estudos com modelos estocásticos apenas para previsão de ΔP na coluna A utilizando duas abordagens: decomposição da série em tendência e sazonalidade, extrapolando ambas, e SARIMA. O autor ainda alterou o domínio da série utilizando o primeiro ponto em um intervalo de 10 minutos, para melhor visualização e aplicação dos métodos.

De Marco (2019), em contrapartida, estudou diretamente a série temporal do fator de resistência ao escoamento na mesma coluna. Ainda, o autor alterou o domínio da série temporal, utilizando os valores da variável observada em função dos ciclos, levando em conta que há vales de variação de pressão (e consequentemente de degradação) a cada fim de ciclo de adsorção (Figura 5.6). Assim, através da derivada de *R* em relação ao tempo, foi possível a distinção entre os períodos de adsorção e regeneração. O autor então estudou 3 instantes diferentes de tempo diferente para cada ciclo:

- 30% do ciclo de adsorção.
- 50% do ciclo de adsorção.
- 75% do ciclo de adsorção.

Figura 5.6 – Padrão cíclico do fator de resistência ao escoamento em TSA.



Fonte: De Marco (2019).

Este trabalho irá aplicar a metodologia com redes neurais LSTM a fim de estudar seu potencial de aplicação em ambas séries temporais, comparando os resultados com os apresentados anteriormente. As séries receberão nome de DP, para a série variação de pressão em função do tempo, e C30, C50 e C75, para as séries degradação em função do ciclo em 30%, 50% e 75% do ciclo, respectivamente.

6 RESULTADOS

6.1 FORNECIMENTO DE GASOLINA A NO RIO GRANDE DO SUL

6.1.1 Modelos estocásticos

Conforme a metodologia, a modelagem do conjunto de treino foi realizada se usando os modelos Holt-Winters e SARIMA. Na Tabela 6.1 constam os coeficientes dos respectivos modelos, respectivamente. A configuração do modelo SARIMA mais adequado para a série VOL é SARIMA (2,1,0)(1,0,1,12), já o modelo Holt-Winters mais adequado foi o modelo com sazonalidade multiplicativa.

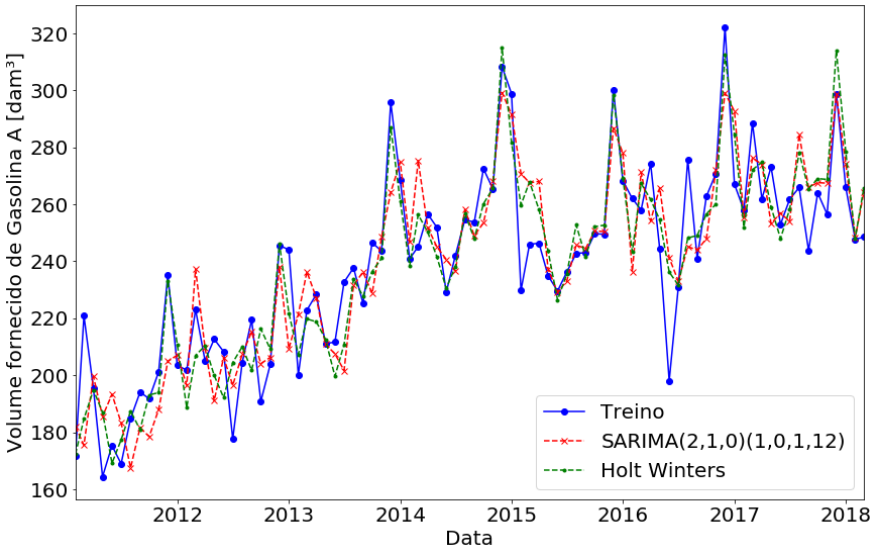
Tabela 6.1 – Parâmetros obtidos para o modelo Holt-Winters.

Modelo	Coeficiente	Valor
Holt-Winters multiplicativo	*a*	0,3276
	c	$2{,}201 \times 10^{-19}$
	d	$6{,}607 \times 10^{-17}$
SARIMA (2,1,0(1,0,1,12)	$\emptyset(1)$	-0,5253
	$\emptyset(2)$	-0,4817
	$\Phi(1^{12})$	0,9631
	$\Theta(1^{12})$	-0,6787
	AIC	**737,886**

Fonte: Elaborado pelo autor (2019).

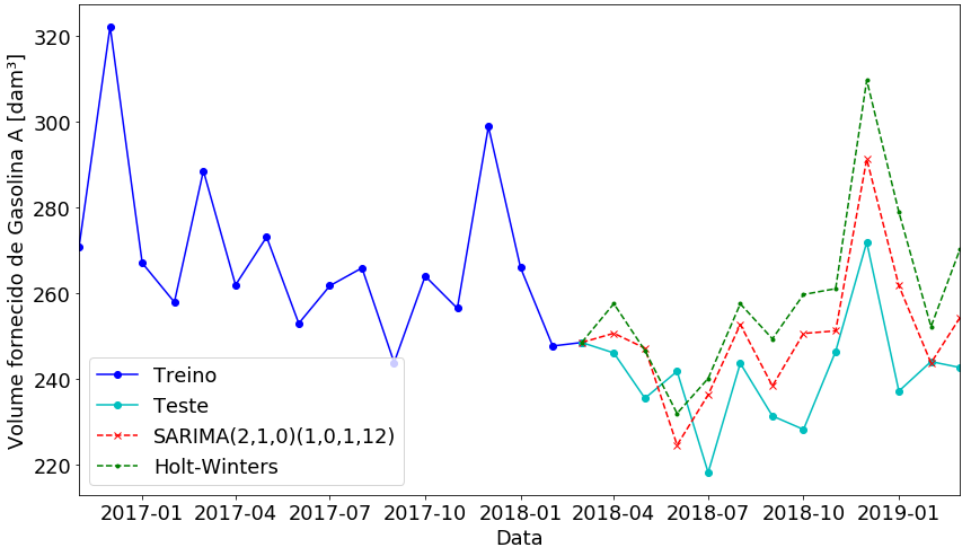
Observando-se a Tabela 6.1, conclui-se que a suavização das componentes de tendência e sazonalidade do modelo Holt-Winters foi desprezível, tendo apenas o coeficiente de suavização de nível um valor considerável. As Figuras 6.1 e 6.2 e a Tabela 6.2 descrevem os resultados obtidos para os ajustes do conjunto de treino e das para as previsões para ambos os modelos, respectivamente.

Figura 6.1 – Ajuste dos modelos para a série de treino para Holt-Winters e SARIMA(2,1,0)(1,0,1,12).



Fonte: Elaborado pelo autor (2019).

Figura 6.2 – Previsões para Holt-Winters e SARIMA(2,1,0)(1,0,1,12).



Fonte: Elaborado pelo autor (2019).

Tabela 6.2 – Resultados obtidos para os ajustes do conjunto de treino e previsões.

Conjunto	Modelo	Resíduo Estacionário	REQM
Treino	Holt-Winters	Não	13,047
Teste	Holt-Winters	Não	23,367
Treino	SARIMA(2,1,0)(1,0,1,12)	Não	16,438
Teste	SARIMA(2,1,0)(1,0,1,12)	Não	14,584

Fonte: Elaborado pelo autor (2019).

É possível concluir que, observando a Tabela 6.2, apesar de apresentar ajuste melhor à série de treino, as previsões geradas pelo método de Holt-Winters são menos representativas do que as obtidas pelo modelo SARIMA, que, por sua vez, apresentou resíduos não estacionários em ambos os casos.

6.1.2 Análise do comportamento dinâmico

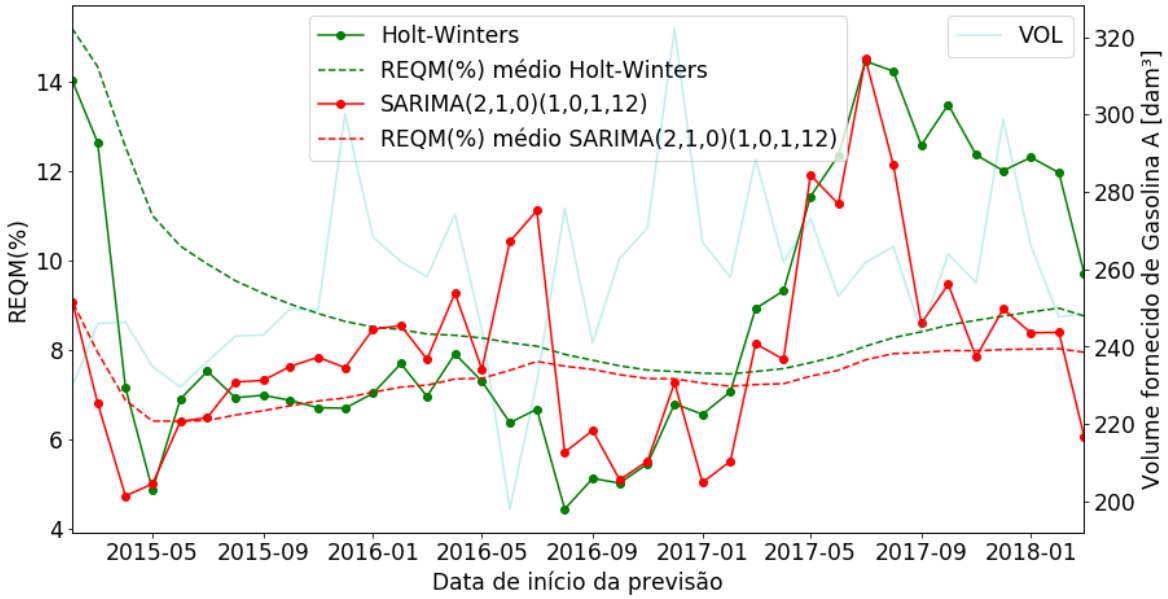
Visando avaliar o potencial de uso dos modelos para previsões em diferentes períodos da série, visto que a mesma possui variações cíclicas de tendência consideráveis, repetiu-se o procedimento de divisão treino-teste ilustrado pela Figura 5.3 para realização de previsões anuais em todos os pontos da série a partir de fevereiro de 2015, com o intuito de avaliar o desempenho dos modelos ao longo do período estudado, bem como sua capacidade de adequação às alterações de comportamento da série temporal.

Levando em conta que os conjuntos de teste para cada ponto de previsão são diferentes, avaliar a REQM por si só é insuficiente para se obter comparações entre os erros de cada previsão. Para isso, usa-se a razão entre a REQM entre as previsões e as suas respectivas séries de teste e média da série de teste, que receberá o nome de REQM(%).

$$REQM(\%) = \left({REQM}/{\overline{Ts_t}} \right) x100 \qquad (6.4)$$

A Figura 6.3 ilustra descreve o REQM(%) obtido para cada previsão iniciada, bem como o valor médio do mesmo até o mês de previsão.

Figura 6.3 - Resultados obtidos alterando-se o ponto de previsão para a série VOL.



Fonte: Elaborado pelo autor (2019).

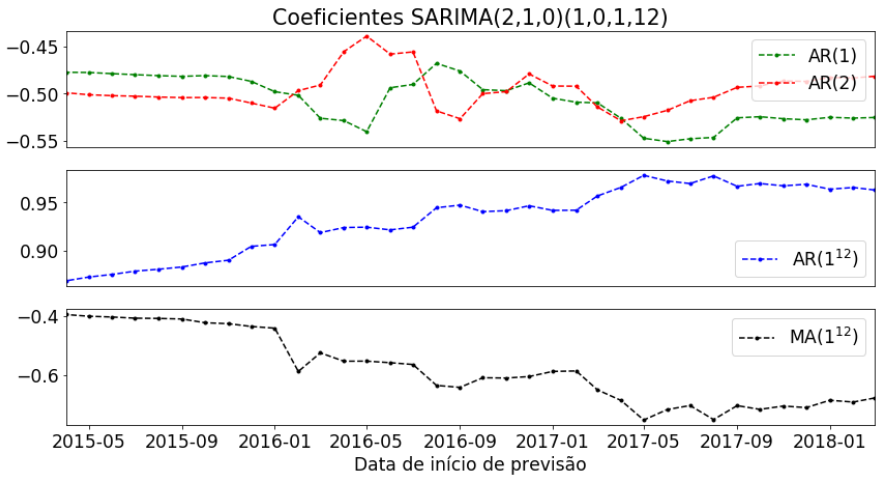
Observa-se pela Figura 6.3 que o REQM(%) aumenta consideravelmente em períodos de alteração de ciclo, tendo seus valores mais altos no início de 2015 e no terceiro trimestre de 2017. Inicialmente, o modelo Holt-Winters supera os ajustes por SARIMA; todavia, à medida em que se aumenta o tamanho do conjunto de treino, o modelo SARIMA ainda apresentou melhor ajuste para a série VOL em relação ao modelo Holt-Winters. Isso se dá pela maior complexidade do modelo SARIMA, que exige por consequência um conjunto de dados maior para realizar análises mais representativas.

Outro ponto de destaque é que houve apenas uma previsão que forneceu resíduos estacionários: Holt-Winters prevista em março de 2016, a qual não foi a que obteve o melhor valor de REQM(%), concluindo-se que este não deve ser o primeiro critério de avaliação para as previsões.

Além de avaliar o comportamento dinâmico através do REQM(%), foi também estudado de maneira análoga os coeficientes dos modelos SARIMA(2,1,0)(1,0,1,12) e Holt-Winters, e como os mesmos variaram ao longo do período, ilustrados pelas Figuras 6.4 e 6.5, respectivamente. Os coeficientes do modelo SARIMA e o coeficiente de suavização de nível do modelo Holt-Winters (*a*) oscilam no período entre fevereiro de 2016 e setembro de 2019. Ainda, observa-se pela Figura 6.3 que o mesmo período é o de maior oscilação do REQM(%) de ambos modelos.

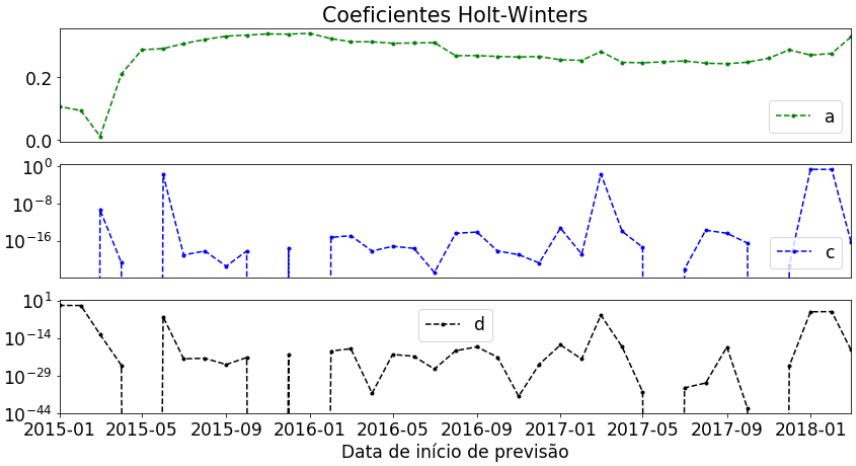
Analogamente aos resultados para previsões recentes, os coeficientes de suavização de tendência (*c*) e sazonalidade (*d*) do modelo Holt-Winters oscilaram entre valores extremamente pequenos e desprezíveis para o modelo.

Figura 6.4 - Coeficientes do modelo SARIMA(2,1,0)(1,01,12) para a série VOL em função do ponto inicial de previsão.



Fonte: Elaborado pelo autor (2019).

Figura 6.5 - Coeficientes do modelo Holt-Winters para a série VOL em função do ponto inicial de previsão.



Fonte: Elaborado pelo autor (2019).

6.1.3 Variável exógena – Dólar americano

Conforme foi constatado no Estudo de Caso, de 2011 a 2014 houve forte correlação entre a série VOL e a cotação de compra do dólar americano. A fim de se avaliar, de fato, o uso do dólar americano como variável exógena poderia acrescentar melhorias significativas a nossas previsões, divide-se o período em questão (janeiro de 2011 a dezembro de 2014) em treino e teste, com o intuito de se analisar somente esse período específico de tempo.

Serão avaliados dois métodos envolvendo variável exógena:

- SARIMAX.
- Regressão Linear em relação à variável exógena seguido de SARIMA nos resíduos.

Salienta-se que, por tratar-se de uma análise de dependência de valores muito antigos, as variáveis exógenas utilizadas serão os valores reais referentes ao período de teste. Ainda,

considerando que há adição de uma nova variável, e que a parte da série VOL estudada possui uma tendência linear contínua, é evidente que os parâmetros p,d,q,P,D e Q devem ser recalculados. A Tabela 6.3 descreve os resultados obtidos para cada método utilizado.

Tabela 6.3 – Configuração e coeficientes obtidos para cada um dos métodos utilizados para previsão da série VOL utilizando variável exógena entre 01/2011 a 12/2014.

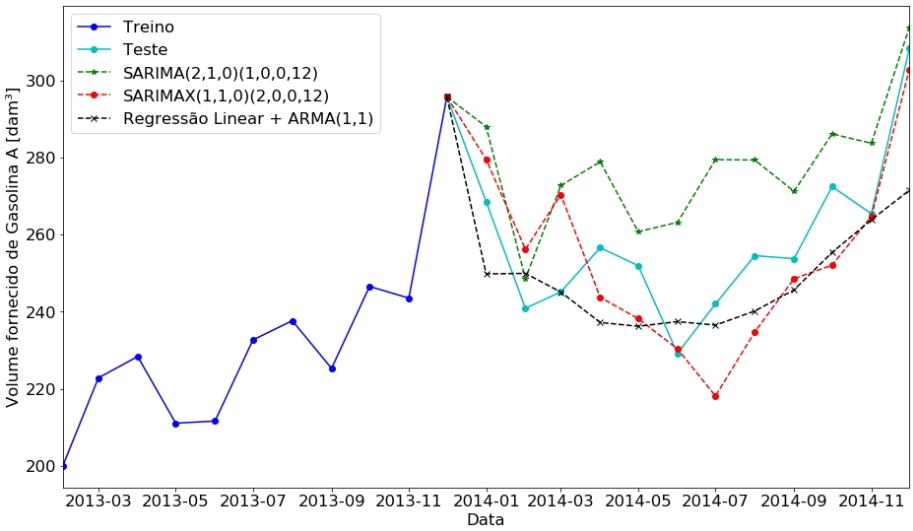
Parâmetro	β_0	β_1	$\emptyset(1)$	$\emptyset(2)$	$\Theta(1)$	$\Phi(1^{12})$	$\Phi(2^{12})$	AIC
SARIMA(2,1,0)(1,0,0,12)	-	-	-0,4888	-0,6069	-	0,5962	-	**307,198**
SARIMAX((1,1,0)(2,0,0,12)	-	107,876	-0,5135	-	0,9195	0,1772	0,4904	**315,093**
RL+ARMA(1,1)	47,997	84,3258	1	-	-0,9941	-	-	**-59,386**

Fonte: Elaborado pelo autor (2019).

O modelo SARIMA que apresentou melhor AIC para os resíduos da regressão linear foi SARIMA (1,0,1)(0,0,0,12), ou seja, um modelo ARMA (1,1). O modelo RL + ARMA foi gerado através da soma da previsão geradas pelo modelo ARMA(1,1) dos resíduos com a gerada pelo modelo de regressão linear exógena.

A Figura 6.6 e a Tabela 6.4 ilustram, respectivamente, o gráfico e os resultados das previsões obtidas com cada um dos métodos supracitados.

Figura 6.6 - Previsões da série VOL para dezembro de 2014, comparando métodos utilizando o dólar americano como variável exógena.



Fonte: Elaborado pelo autor (2019).

Tabela 6.4 – Resultados obtidos para as previsões de dezembro de 2014 utilizando dólar americano como variável exógena.

Modelo	Resíduo Estacionário	REQM
SARIMA(2,1,0)(1,0,0,12)	Não	21,979
SARIMAX(1,1,0)(2,0,0,12)	Não	15,224
Reg. Linear + ARMA(1,1)	Não	16,003

Fonte: Elaborado pelo autor (2019).

A Tabela 6.4 nos mostra que, de fato, a inserção da variável exógena possibilitou uma melhora nas previsões, com ambos os métodos apresentando REQM semelhantes, com uma redução de em torno de 30% do REQM do modelo estocástico puramente endógeno. Apesar do erro próximo, vemos pela Figura 6.6 que o comportamento de ambos os modelos diverge um do outro. Enquanto o SARIMAX tem seu erro derivado de diferenças em nível com o conjunto de teste, possuindo um ajuste aceitável em seu comportamento, o erro do modelo que combina regressão linear com ARMA se dá pela incapacidade do modelo de simular as oscilações da série original. Dito isso, para previsões médias, ambos os modelos serviriam no período estudado, mas deixariam a desejar em previsões pontuais.

6.1.4 Redes Neurais LSTM

A fim de se obter o melhor ajuste possível para a série temporal, foram comparados diversos valores diferentes referentes a número de neurônios e número de épocas intuitivamente. O valor de dados de entrada a serem alimentados para gerar um ponto (*window*) foi fixado em 24, equivalente a dois períodos sazonais e aproximadamente 28% do conjunto de treino. De todas arquiteturas testadas, a que apresentou melhores resultados foi a LSTM (35,24,300).

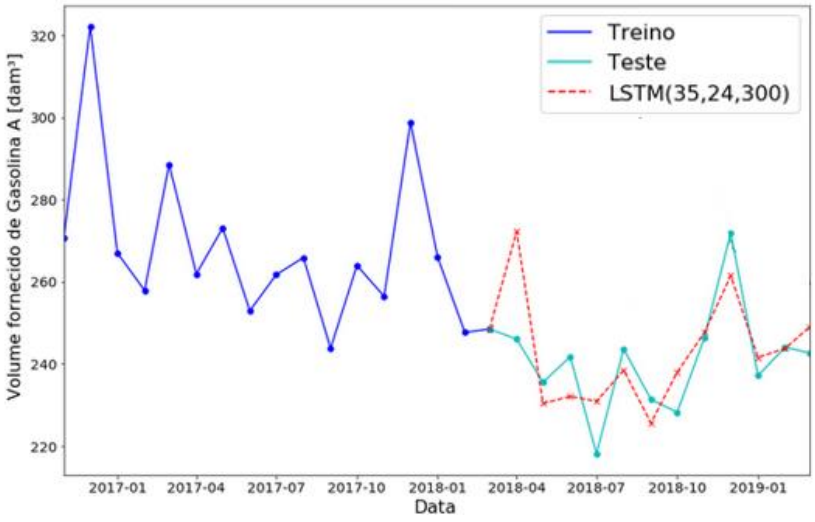
Conforme a metodologia, escolheu-se o método estocástico de melhor desempenho para o ajuste da série de treino preliminar às previsões. No atual caso, aplicou-se um ajuste por SARIMA (2,1,0)(1,0,1,12) à série de treino e repetiu-se o processo de previsão por LSTM (35,24,300), conforme mostram a Tabela 6.5 e a Figura 6.7.

Tabela 6.5 – Resultados obtidos para LSTM (35,24,300) para a série VOL.

Modelo	Resíduo Estacionário	REQM
LSTM	Sim	10,369

Fonte: Elaborado pelo autor (2019).

Figura 6.7 - Previsões realizadas pelo método de redes neurais LSTM(35,24,300) para 12 meses da série VOL.



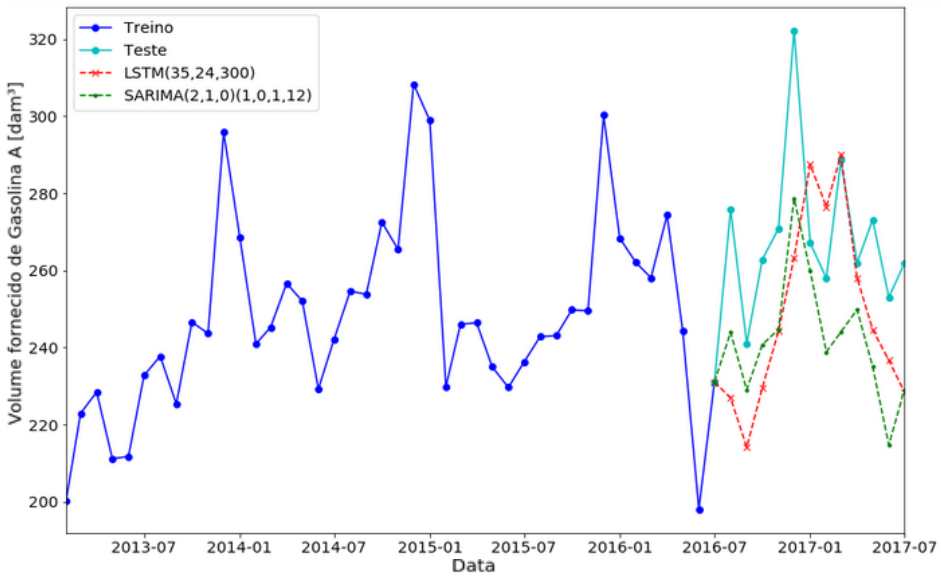
Fonte: Elaborado pelo autor (2019).

O ajuste por LSTM mostrou alto potencial para a série VOL, diminuindo o REQM em relação ao modelo SARIMA em 28,90%, além de obter resíduos estacionários, mesmo obtendo uma previsão para o primeiro ponto consideravelmente equivocada (facilmente

observável na Figura 6.7). O tempo de treino da rede neural foi de aproximadamente 300 segundos.

Por fim, afim de comparar a aplicabilidade do método na série VOL em detrimento do modelo SARIMA, aplicou-se a mesma configuração de rede neural LSTM ao ponto de previsão cujo REQM(%) foi o maior para os modelos estocásticos: julho de 2017. A Figura 6.8 e a Tabela 6.6 descrevem respectivamente as previsões e os resultados.

Figura 6.8 - Previsões para a série VOL a partir de julho de 2017 utilizando redes neurais LSTM e SARIMA.



Fonte: Elaborado pelo autor (2019).

Tabela 6.5 – Resultados obtidos para a série VOL a partir de julho de 2017 utilizando SARIMA e Redes Neurais LSTM.

Modelo	Resíduo Estacionário	REQM
SARIMA	Não	36,669
LSTM	Não	26,868

Fonte: Elaborado pelo autor (2019).

É possível concluir pela Figura 6.8 que o modelo obtido por redes neurais é mais capaz de "recuperar-se" de um ponto atípico, como o ponto de vale anterior, muito abaixo dos mínimos sazonais anteriores, do que o modelo SARIMA, apesar do último apresentar um comportamento mais próximo ao da série original. Dito isso, analisando o REQM obtido pelo modelo LSTM, 26,73% menor que REQM referente ao modelo SARIMA, considera-se os resultados um aprimoramento. Logicamente, por possuir menos pontos, o tempo de treino da rede neural foi inferior ao correspondente aos ajustes para dados recentes, tendo um valor de aproximadamente 130 segundos.

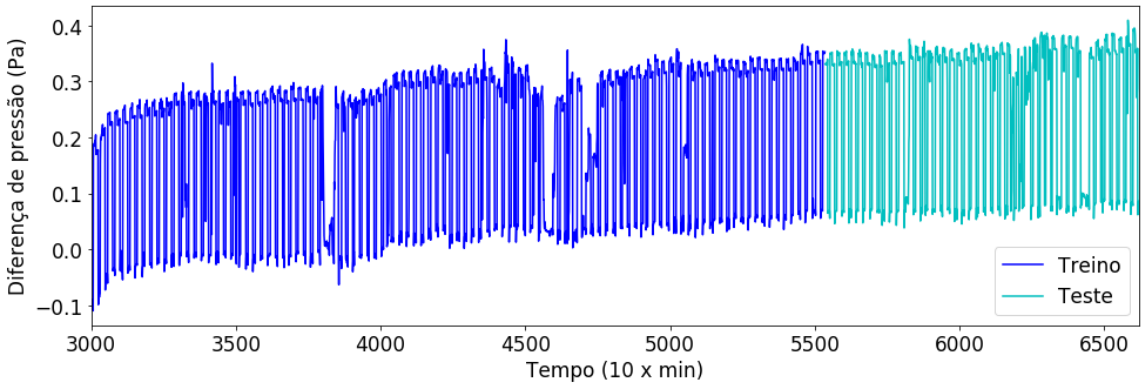
6.2 UNIDADE TSA PARA DESIDRATAÇÃO DE GÁS NATURAL

6.2.1 Diferença de pressão em função do tempo

Analogamente a Favarini (2018), escolheu-se o primeiro ponto a cada dez minutos para a análise da série temporal. A série original foi então tratada removendo-se os períodos

correspondentes a etapas de regeneração da coluna. Começando-se a partir do mesmo ponto que o autor, obteve-se uma série temporal contendo 3623 pontos, dos quais 2563 formaram o conjunto de treino e, consequentemente, 1087 o conjunto de teste, como mostra a Figura 6.9. Para a série DP, não foi preciso a normalização dos dados, visto que os valores da série já são pequenos o suficiente.

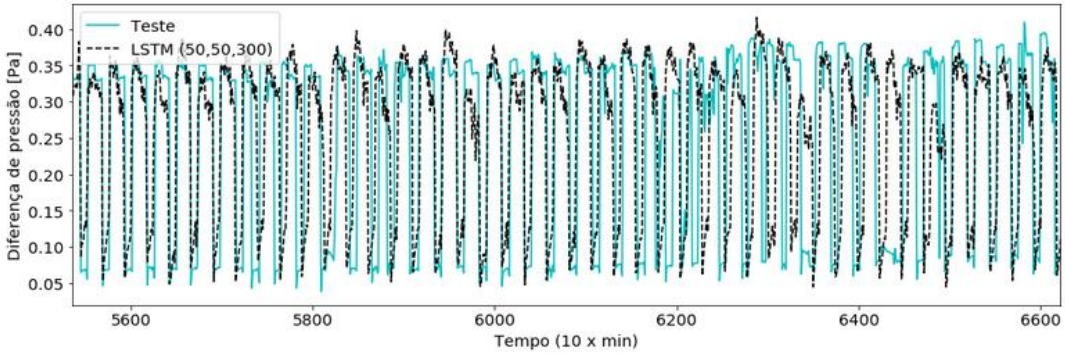
Figura 6.9 - Divisão treino-teste da série DP.



Fonte: Elaborado pelo autor (2019).

A arquitetura da rede neural LSTM mais adequada foi LSTM (50,50,300), ou seja, contendo 50 neurônios, utilizando 50 pontos como dado de entrada e com 300 épocas de treino, o qual durou aproximadamente 230 minutos. A Figura 6.10 ilustra as previsões, e a Tabela 6.7 compara os resultados aos obtidos por Favarini (2018).

Figura 6.10 - Ajuste para a série DP utilizando LSTM (50,50,300).



Fonte: Elaborado pelo autor (2019).

Tabela 6.6 – Comparação entre resultados obtidos para diferença de pressão utilizando LSTM(50,50,300) e resultados da literatura.

Modelo	R^2	REQM	Resíduo Estacionário
Decomposição*	0,782	-	Sim
SARIMA*	0,762	-	Sim
LSTM (50,50,300)	0,08424	0,1275	Sim

Fonte: Elaborado pelo autor (2019); *Favarini (2018).

O resultado obtido foi satisfatório, porém não apresentou uma melhoria significativa a modelos estocásticos utilizados por Favarini (2018), especialmente devido ao alto tempo requerido para o treino da rede neural.

6.2.2 Degradação em função do número de ciclos

Neste caso, foram analisadas as mesmas três etapas estudadas por De Marco (2019), 30%, 50% e 75% do ciclo de adsorção. A divisão treino-teste foi a mesma utilizada pelo autor, dividindo as séries (todas com períodos correspondentes a 160 ciclos) em um conjunto de treino e teste de 100 e 60 ciclos, respectivamente.

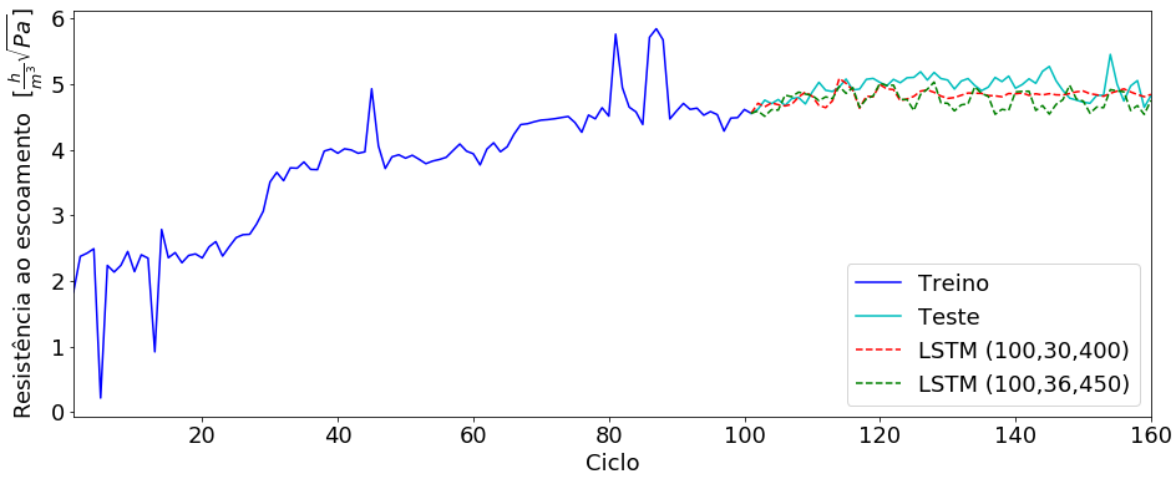
Duas arquiteturas de rede neural LSTM forneceram os melhores resultados para as três séries, ambas contendo 100 neurônios. A primeira consiste na previsão de pontos futuros a partir de 30 pontos anteriores, com 400 épocas de treino, enquanto a segunda utiliza 36 pontos para previsões futuras e 450 épocas. O tempo médio de treino necessário para ambas as configurações foi de, respectivamente, 640 e 770 segundos, não variando significativamente para cada etapa de adsorção. A Tabela 6.8 ilustra os resultados obtidos pelos modelos, comparando-os com os resultados obtidos por De Marco (2019) para o modelo SARIMA, enquanto as Figuras 6.11, 6.12 e 6.13, as previsões.

Tabela 6.7 – Comparação entre resultados obtidos para todas etapas do ciclo de adsorção utilizando LSTM (100,20,400) e LSTM (100,36,450) e resultados da literatura.

Etapa / Modelo	30%		50%		75%	
	REQM	Res. Estacionário	REQM	Res. Estacionário	REQM	Res. Estacionário
LSTM (100,30,400)	0,2076	Sim	0,1930	Sim	0,3964	Sim
LSTM (100,36,450)	0,2764	Não	0,2453	Sim	0,1736	Sim
SARIMA*	0,3047	Não	0,2788	Não	0,3936	Não

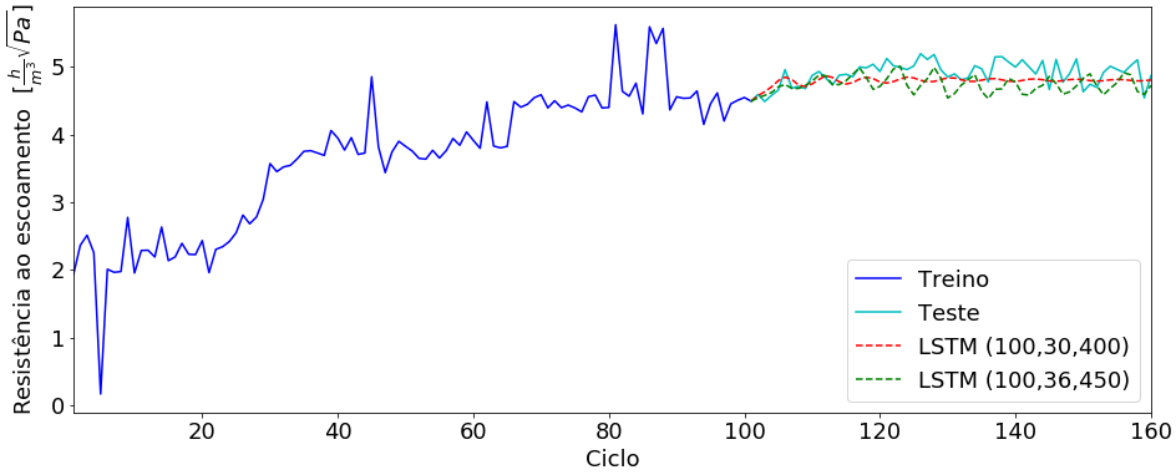
Fonte: Elaborado pelo autor (2019); *De Marco (2019).

Figura 6.11 - Modelos LSTM ajustados para 30% do ciclo de adsorção.



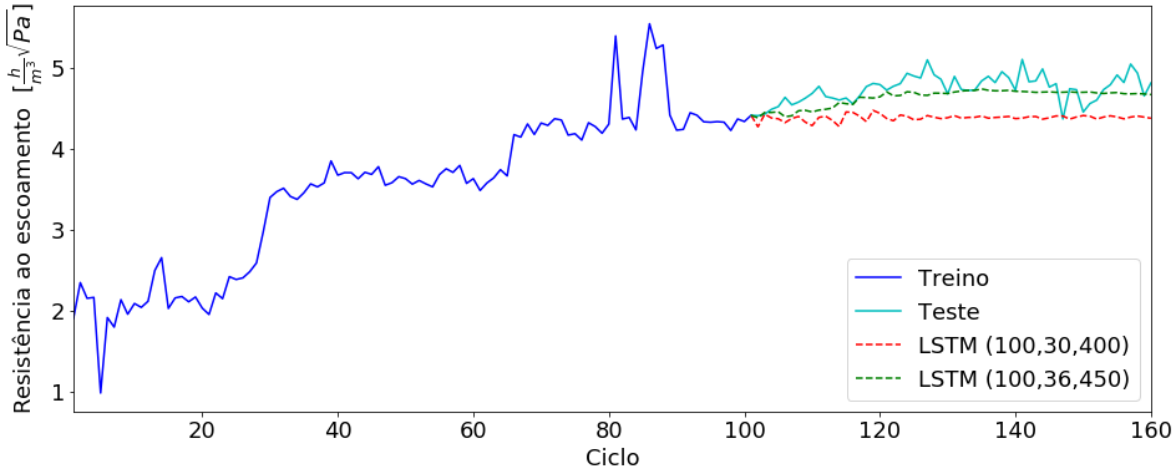
Fonte: Elaborado pelo autor (2019).

Figura 6.12 - Modelos LSTM ajustados para 50% do ciclo de adsorção.



Fonte: Elaborado pelo autor (2019).

Figura 6.13 - Modelos LSTM ajustados para 75% do ciclo de adsorção.



Fonte: Elaborado pelo autor (2019).

Analisando-se as Figuras 6.11 a 6.13, percebe-se que, com exceção do ajuste por LSTM (100,30,400) para a série C75, todos os ajustes conseguem capturar adequadamente a tendência das séries, o que é fundamental para previsões que visam antecipar o atingimento de um valor máximo. Entretanto, nenhum ajuste captou de maneira adequada o comportamento oscilatório da série, especialmente a longo prazo. Ainda, analisando a Tabela 6.8, percebe-se que, excluindo-se novamente o ajuste LSTM (100,30,400) para C75, todos os ajustes obtiveram REQM inferior aos ajustes por SARIMA, além da grande maioria ter obtido previsões com resíduo estacionário.

7 CONCLUSÕES E SUGESTÕES PARA TRABALHOS FUTUROS

No presente trabalho, estudou-se dois casos distintos, constituídos por séries temporais cujas dinâmicas diferem consideravelmente. Para o fornecimento de Gasolina A no Rio Grande do Sul, avaliou-se o potencial de aplicação de modelos estocásticos e redes neurais LSTM para predição de um ano, bem como a confiabilidade dos mesmos em diferentes períodos da atual década. No tocante do estudo de séries temporais envolvidas no processo TSA em uma unidade de desidratação de gás natural, procurou-se otimizar previsões já estudadas utilizando o método de redes neurais LSTM, comparando os resultados obtidos neste trabalho com os obtidos pelos seus respectivos autores.

A escolha dos modelos estocásticos utilizados foi feita através da aplicação do critério de informação AIC, para os modelos SARIMA, e do método dos mínimos quadrados, para o método de Holt-Winters. Para a escolha da configuração das redes neurais, foram testadas diversas combinações de valores para número de neurônios, dados de entrada e épocas.

Para o primeiro caso, entre os modelos estocásticos, destacou-se a aplicação do modelo SARIMA, com uma configuração SARIMA (2,1,0)(1,0,1,12), em detrimento do modelo Holt-Winters, ainda que as previsões obtidas não foram confiáveis para uma previsão ponto-a-ponto. O primeiro ainda prevaleceu, de modo geral, ao longo da década, porém não apresentando aplicação confiável, visto que, em períodos próximos a mudanças de ciclos, os valores de REQM obtidos foram consideravelmente elevados, chegando a atingir valores superiores a 14% do valor da média da série de teste. Ainda, constatou-se que, na parcela inicial do conjunto de dados, existe correlação entre a quantidade de Gasolina A fornecida no estado do RS com a cotação de compra do dólar americano, porém a mesma se perdeu a partir de 2015.

No caso das previsões de fornecimento de Gasolina A no RS, a aplicação de redes neurais LSTM mostrou-se consideravelmente positiva, obtendo resultados com REQM 28,90% inferiores aos obtidos com SARIMA. A comprovação da aplicabilidade do método na série temporal em detrimento de modelos estocásticos se deu pela aplicação do mesmo no ponto cujo REQM(%) foi o maior para os modelos estocásticos, julho de 2017, apresentando um REQM 26,81% inferior ao obtido pelo modelo SARIMA.

Para o segundo caso, estudou-se duas diferentes séries constituintes do mesmo processo. Para a série temporal da diferença de pressão em uma coluna de adsorção, constatou-se que a aplicação de redes neurais LSTM, cuja melhor arquitetura obtida foi LSTM (50,50,300), não forneceu resultados tão superiores aos obtidos por modelos estocásticos na literatura para justificar a aplicação de um método de tal complexidade. Todavia, estudando-se séries de degradação em função do número de ciclos em três etapas distintas do ciclo de adsorção – 30%, 50% e 75% –, constatou-se um forte potencial de aplicação dessa ferramenta para essa área de estudo.

As arquiteturas das redes neurais LSTM aplicadas à série de degradação da coluna foram LSTM (100,30,400) e LSTM (100,36,450), a primeira obteve erros inferiores para todas etapas em detrimento de ajustes por SARIMA, enquanto a segunda, com exceção do ajuste para 75% do ciclo de adsorção, que obteve desempenho muito aquém do esperado, obteve erros ainda menores.

Para trabalhos futuros, pode-se sugerir o aprofundamento da utilização de redes neurais para ambos os casos, aplicando-se combinações de redes neurais LSTM com redes neurais recorrentes bidirecionais (BRNN), que consistem na utilização de valores futuros para predição da série temporal. Ainda, estudos detalhados sobre variáveis exógenas da série temporal de fornecimento de Gasolina A podem ser conduzidos, visando se compreender a dependência ocasional da série em relação a cotação do dólar americano, bem como a identificação de novas variáveis exógenas.

REFERÊNCIAS

Agência Nacional de Petróleo, Gás Natural e Biocombustíveis (ANP). **Seminário de Avaliação do Mercado de Combustíveis 2018**, 2018. Disponível em < http://www.anp.gov.br/images/Palestras/seminario_avaliacao_2018.pdf>. Acesso em 30 mai. 2019.

Agência Nacional de Petróleo, Gás Natural e Biocombustíveis (ANP). **Dados de Mercado**, 2019a. Disponível em <http://www.anp.gov.br/distribuicao-e-revenda/distribuidor/combustiveis-liquidos/dados-de-mercado>. Acesso em 25 abr. 2019.

Agência Nacional de Petróleo, Gás Natural e Biocombustíveis (ANP). **Gás natural**, 2019b. Disponível em <http://www.anp.gov.br/gas-natural>. Acesso em 27 mai. 2019.

BAFFA, A. C. E.; SOARES, J. A.; GOLDSCHMIDT, R. R. Previsão de Séries Temporais Utilizando Redes Neurais Artificiais, In: INTERNATIONAL CONFERENCE ON INFORMATION SYSTEMS AND TECHNOLOGY MANAGEMENT, 5., 2008, São Paulo. **Anais** ... São Paulo: [s.n]. p. 142-163, 2008.

BANCO CENTRAL. **Sistema Gerenciador de Séries Temporais (SGS)**, 2019. Disponível em <https://www3.bcb.gov.br/sgspub/>. Acesso em 3 mai 2019.

BARUQUE, Bruno et al. Geothermal heat exchanger energy prediction based on time series and monitoring sensors optimization. **Energy**, v. 171, p. 49-60, 2019.

BOX, G. E. P.; JENKINS, G. M.; REINSEL, G. C. **Time Series Analysis**. Forecasting Control. 5. ed. Hoboken: John Wiley & Sons, 2015.

BROCKWELL, P. J.; DAVIS, R. A. **Introduction to Time Series and Forecasting**. 3. ed. New York: Springer, 2006.

CASTELÃO, R. **Utilização de Redes Neurais para Previsões no Mercado de Ações**. 2018. Relatório Técnico (Graduação em Ciência da Computação) - Instituto de Computação, Universidade Estadual de Campinas, Campinas, 2018.

CAO, Jian.; LI, Zhi.; LI, Jian. Financial time series forecasting model based on CEEMDAN and LSTM, **Physica A: Statistical Mechanis and its Applications**, v. 519, p. 127-139, 2019.

CHEN, Edwin. **Exploring LSTMs**. 2017. Disponível em: <http://blog.echen.me/2017/05/30/exploring-lstms/>. Acesso em 19 mai. 2019.

DE MARCO, L. M. **Determinação do Tempo de Vida Útil Remanescente em Processos Cíclicos**. 2019. Dissertação (mestrado em Engenharia Química) - Escola de Engenharia, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2019.

EHLERS, R. S. **Análise de Séries Temporais**. 4. ed. Curitiba: Ricardo Sanders Ehlers, 2017.

FAVARINI, T. O. **Análise de Séries Temporais: Comparação entre modelos preditivos em estudo de caso**. 2018. Trabalho de Conclusão (Graduação em Engenharia Química) - Escola de Engenharia, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2018. Disponível em: <https://lume.ufrgs.br/handle/10183/193141>. Acesso em 30 maio 2019.

FONSECA, N. A. **Simulação do processo de adsorção PSA para separação da mistura etanol-água**. 2011. Dissertação (Mestrado em Engenharia Química) - Faculdade de Engenharia Química, Universidade Estadual de Campinas, Campinas, 2011.

GLOROT, Xavier; BENGIO, Yoshua. Understanding the difficulty of training deep feedforward neural networks. In: International Conference on Artificial Intelligence and Statistics, 2010. **Anais** ... [s.l:s.n]

GUESMI, Latifa; FATHALLAH, Habib; MENIF, Mourad. Modulation Format Recognition Using Artificial Neural Networks for the Next Generation Optical Networks. In: Advanced Applications for Artificial Neural Networks. In: EL-SHAHAT, Adel. **Advanced Applications for Artificial Neural Networks**. 1. ed. [s.l]: InTech, 2018. cap. 2, p. 12–27.

GUIDOLIN, Massimo; PEDIO, Manuela. **Essentials of Time Series for Financial Applications***.* 1. ed. Cambridge: Academic Press, 2018.

KINGMA, D. P.; BA, J. L. Adam: a method for stochastic optimization. In: International Conference on Learning Representations, 2015. **Anais** ... [s.l:s.n]

LIMA, M. B. S. P. et al. Aplicação do modelo de previsão de demanda Holt-Winters em uma regional de corte e dobra de aço. In: ENCONTRO NACIONAL DE ENGENHARIA DE PRODUÇÃO, 15., 2015, Fortaleza. **Anais** ... Fortaleza: ABEPRO, 2015. p. 2-21.

LUSTOSA, Leonardo et al. **Planejamento e Controle da Produção**. 4. ed. Rio de Janeiro: Elsevier Editora, 2008.

MAÇAIRA, P. M. et al. Time series analysis with explanatory variables: A systematic literature review*,* **Environmental Modelling & Software***,* v. 107, p. 199-209, 2018.

MIRANDA, C. V. C. et al. Previsão de dados de alta frequência para carga elétrica usando Holt-Winters com dois ciclos. In: SBPO, 41., 2009, Porto Seguro. **Anais** ... [s.l.: s.n.], 2013. p. 222-232.

MORETTIN, P. A.; TOLOI, C. M. C. **Análise de séries temporais**. 2. ed. São Paulo: Edgard Blucher, 2006.

MOUSAVI, M. H.; GHAVIDEL, Saleh. Structural time series model for energy demand in Iran's transportation sector*,* **Case Studies on Transport Policy**, v. 7, n. 2, p. 423-432, 2019.

NELDER, J. A.; WEDDERBURN R. W. M. Generalized Linear Models*,* **Journal of the Royal Statistical Society***,* v. 135, n. 3, p. 370-384, 1972.

NELSON, D. M. Q. **Uso de redes neurais recorrentes para previsão de séries temporais financeiras**. 2017. Dissertação (Mestrado em Ciência da Computação) - Instituto de Ciências Exatas, Universidade Federal de Minas Gerais, Belo Horizonte, 2017.

NEUSSER, Klaus. **Time Series Econometrics**. 1. ed. New York: Springer, 2016.

OLAH, Christopher. **Understanding LSTM Networks**, 2015. Disponível em: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>. Acesso em 19 jun. 2019.

ORDÓÑEZ, Celestino et al. A hybrid ARIMA-SVM model for the study of the remaining useful life of aircraft engines, **Journal of Computational and Applied Mathematics**, v. 436, p. 184-191, 2019.

OZOEGWU, C. G. Artificial neural network forecast of monthly mean daily global solar radiation of selected locations based on time series and month number, **Journal of Cleaner Production**, v. 216, p. 1-13, 2019.

PAL, Avishek; PRAKASH, P. K. S**. Practical Time Series Analysis**: Master Time Series Data Processing, Visualization, and Modeling using Python. Birmingham: Packt Publishing, 2017.

PORTILLA, José. **Python for Time Series Data Analysis**, Udemy. 2019. Disponível em: <https://www.udemy.com/python-for-time-series-data-analysis/>. Acesso em 28 mai. 2019.

SAGHEER, Alaa.; KOTB, Mostafa. Time series forecasting of petroleum production using deep LSTIM recurrent networks, **Neurocomputing**, v. 323, p. 203-213. 2019.

SUHARTONO; LEE, M. H.; PRASTYO, D. D. Two levels ARIMAX and regression models for forecasting time series data with calendar variation effects. In: AIP CONFERENCE PROCEEDINGS 2015. **Anais** ... [s.l]: AIP Publishing.

TAMURA, L. K. **Séries temporais com variáveis exógenas e gráficos de controle como ferramentas de decisão no mercado financeiro**. 2019. Trabalho de conclusão (Graduação em Engenharia de Produção) - Escola Politécnica, Universidade Federal de São Paulo, São Paulo, 2019.

TAŞPINAR, Fatih; NUMAN, Çelebi; TUTKUN, Nedim. Forecasting of daily natural gas consumption on regional basis in Turkey using various computational methods, **Energy and Buildings**, v. 56, p. 23-31, 2013.

TIBULO, Cleiton. **Modelos de séries temporais aplicados a dados de umidade relativa do ar**. 2019. Dissertação (Mestrado em Engenharia de Produção) - Centro de Tecnologia, Universidade Federal de Santa Maria, Santa Maria, 2014.

WALTER, O. M. G. C. et al. Aplicação de um modelo SARIMA na previsão de vendas de motocicletas, **Exacta**, v. 11, n. 1, pp. 77-88, 2013.

XAVIER, Jorge. **Análise e previsão de séries temporais com modelos ARIMA e análise espectral singular**. 2016. Dissertação (Mestrado em Bioestatística e Biometria) - Universidade Aberta, Lisboa, 2016.

Apêndice A – Resultados da análise do comportamento dinâmico para a série VOL

Tabela 0.1 – Resultados da análise de comportamento dinâmico para o modelo SARIMA (VOL).

Ponto de previsão	RMEQ	Média	Resíduo Estacionário	Média Testes	ø(1)	ø(2)	Φ(1^12)	Θ(1^12)	RMEQ(%)
mar-18	14,584	250,248	False	240,604	-0,5253	-0,4817	0,9631	-0,6787	6,061
fev-18	20,227	258,266	False	241,090	-0,5259	-0,4835	0,9657	-0,6924	8,390
jan-18	20,223	258,122	False	241,393	-0,5251	-0,4829	0,9639	-0,6865	8,378
dez-17	21,721	262,603	False	243,790	-0,5277	-0,4871	0,9691	-0,7112	8,910
nov-17	19,293	261,774	False	246,036	-0,5265	-0,4866	0,9674	-0,7059	7,841
out-17	23,351	267,201	False	246,872	-0,5244	-0,4919	0,9699	-0,7172	9,459
set-17	21,460	268,410	False	249,851	-0,5256	-0,4934	0,9669	-0,7045	8,589
ago-17	30,453	279,356	False	250,890	-0,5463	-0,5039	0,9778	-0,7517	12,138
jul-17	36,669	287,485	False	252,743	-0,5477	-0,5076	0,9698	-0,7042	14,508
jun-17	28,860	281,600	False	256,371	-0,5507	-0,5177	0,9725	-0,7177	11,257
mai-17	30,633	283,767	False	257,303	-0,5472	-0,5244	0,9784	-0,7532	11,905
abr-17	20,273	272,239	False	260,436	-0,5261	-0,5287	0,9658	-0,6869	7,784
mar-17	21,297	276,659	False	261,756	-0,5098	-0,5141	0,9569	-0,6511	8,136
fev-17	14,609	269,194	False	265,092	-0,5091	-0,4922	0,9419	-0,5872	5,511
jan-17	13,382	267,063	False	265,945	-0,5049	-0,4920	0,9418	-0,5887	5,032
dez-16	19,329	279,603	False	266,036	-0,4886	-0,4792	0,9466	-0,6061	7,266
nov-16	14,773	266,597	False	267,981	-0,4969	-0,4980	0,9415	-0,6112	5,513
out-16	13,704	267,274	False	269,172	-0,4958	-0,5000	0,9405	-0,6101	5,091
set-16	16,674	257,601	False	269,072	-0,4762	-0,5266	0,9473	-0,6428	6,197
ago-16	15,346	258,249	False	268,836	-0,4680	-0,5183	0,9445	-0,6361	5,708
jul-16	29,964	242,325	False	269,654	-0,4903	-0,4559	0,9242	-0,5653	11,112
jun-16	27,832	243,292	False	267,091	-0,4938	-0,4583	0,9214	-0,5592	10,420
mai-16	19,855	266,038	False	262,512	-0,5402	-0,4392	0,9242	-0,5539	7,564
abr-16	24,103	276,220	False	260,112	-0,5286	-0,4556	0,9239	-0,5542	9,266
mar-16	20,332	263,962	False	261,142	-0,5260	-0,4909	0,9186	-0,5260	7,786
fev-16	22,085	269,555	False	258,600	-0,5018	-0,4969	0,9351	-0,5890	8,540
jan-16	21,906	248,997	False	258,939	-0,4980	-0,5154	0,9061	-0,4424	8,460
dez-15	19,685	254,867	False	259,047	-0,4871	-0,5100	0,9042	-0,4366	7,599
nov-15	20,149	247,956	False	257,216	-0,4819	-0,5048	0,8897	-0,4272	7,833
out-15	19,486	247,343	False	255,440	-0,4810	-0,5041	0,8869	-0,4240	7,628
set-15	18,606	249,486	False	254,351	-0,4818	-0,5042	0,8827	-0,4114	7,315
ago-15	18,524	249,522	False	254,519	-0,4810	-0,5036	0,8803	-0,4094	7,278
jul-15	16,330	249,699	False	251,772	-0,4800	-0,5027	0,8783	-0,4086	6,486
jun-15	16,133	249,844	False	252,203	-0,4789	-0,5022	0,8749	-0,4047	6,397
mai-15	12,745	248,993	False	254,833	-0,4777	-0,5012	0,8721	-0,4020	5,001
abr-15	12,018	250,700	False	254,048	-0,4775	-0,4991	0,8682	-0,3963	4,730
mar-15	17,142	264,853	False	251,720	-0,4960	-0,5841	0,8678	-0,3857	6,810
fev-15	22,762	271,029	False	250,713	-0,5370	-0,5933	0,8913	-0,4518	9,079

Fonte: Elaborado pelo autor (2019).

Tabela 0.2 – Resultados da análise de comportamento dinâmico para o modelo Holt-Winters (VOL).

Ponto de previsão	RMEQ	Média	Resíduo Estacionário	Média Testes	a	c	d	RMEQ(%)
mar-18	23,367	259,602	False	240,604	0,32761	2,201E-19	6,607E-17	9,712
fev-18	28,830	266,297	False	241,090	0,27525	3,528E-04	2,221E-01	11,958
jan-18	29,696	266,546	False	241,393	0,26991	3,101E-04	2,296E-01	12,302
dez-17	29,263	269,749	False	243,790	0,28683	9,734E-26	7,146E-22	12,003
nov-17	30,404	274,322	False	246,036	0,26016	-6,939E-18	0,000E+00	12,358
out-17	33,279	277,965	False	246,872	0,24764	1,249E-42	3,790E-17	13,480
set-17	31,417	279,049	False	249,851	0,24236	3,266E-18	4,737E-15	12,574
ago-17	35,694	284,762	False	250,890	0,24446	1,629E-32	2,185E-14	14,227
jul-17	36,537	287,064	False	252,743	0,25146	1,911E-34	1,008E-22	14,456
jun-17	31,652	284,210	False	256,371	0,24866	-6,939E-18	0,000E+00	12,346
mai-17	29,401	280,882	False	257,303	0,24569	5,096E-36	4,791E-18	11,427
abr-17	24,266	275,252	False	260,436	0,24700	3,482E-18	1,264E-14	9,317
mar-17	23,354	276,466	False	261,756	0,28141	1,456E-05	2,498E-02	8,922
fev-17	18,718	272,352	False	265,092	0,25325	5,196E-23	1,641E-19	7,061
jan-17	17,439	269,494	False	265,945	0,25519	2,129E-17	6,050E-14	6,557
dez-16	18,068	274,210	False	266,036	0,26565	4,182E-25	2,147E-21	6,791
nov-16	14,609	270,703	False	267,981	0,26387	8,677E-38	1,243E-19	5,451
out-16	13,504	266,771	False	269,172	0,26575	3,080E-22	8,011E-19	5,017
set-16	13,788	264,040	False	269,072	0,26852	3,536E-18	8,601E-15	5,124
ago-16	11,930	267,305	False	268,836	0,26807	1,075E-19	4,733E-15	4,438
jul-16	17,987	256,766	False	269,654	0,30948	5,250E-27	2,000E-23	6,670
jun-16	17,012	255,995	False	267,091	0,30847	6,635E-22	2,818E-18	6,370
mai-16	19,148	271,114	False	262,512	0,30761	3,445E-21	7,793E-18	7,294
abr-16	20,569	272,775	False	260,112	0,31222	1,363E-36	7,473E-19	7,908
mar-16	18,162	267,471	True	261,142	0,31245	6,597E-19	1,459E-15	6,955
fev-16	19,889	269,933	False	258,600	0,32227	7,879E-20	6,235E-16	7,691
jan-16	18,191	259,818	False	258,939	0,33924	-6,939E-18	0,000E+00	7,025
dez-15	17,339	261,697	False	259,047	0,33648	3,404E-21	3,342E-18	6,693
nov-15	17,242	259,965	False	257,216	0,33731	-6,939E-18	0,000E+00	6,703
out-15	17,546	260,268	False	255,440	0,33344	2,121E-22	7,682E-19	6,869
set-15	17,762	259,890	False	254,351	0,33013	3,367E-25	4,255E-22	6,983
ago-15	17,633	261,467	False	254,519	0,31956	1,068E-22	7,661E-19	6,928
jul-15	18,934	263,309	False	251,772	0,30663	6,408E-23	1,073E-19	7,520
jun-15	17,400	261,375	False	252,203	0,29063	2,971E-06	1,767E-02	6,899
mai-15	12,398	263,342	False	254,833	0,28712	-6,939E-18	0,000E+00	4,865
abr-15	18,161	269,701	False	254,048	0,20969	1,850E-25	2,452E-21	7,149
mar-15	31,814	282,285	False	251,720	0,01196	3,022E-13	4,709E-10	12,639
fev-15	35,123	284,585	False	250,713	0,09397	9,397E-02	0,000E+00	14,009
jan-15	40,542	287,971	False	248,029	0,10636	1,064E-01	0,000E+00	16,346

Fonte: Elaborado pelo autor (2019).

Apêndice B – Códigos utilizados em Python

CÓDIGOS - FORNECIMENTO DE GASOLINA A NO RIO GRANDE DO SUL

In [ ]:
```
#Importando as bibliotecas
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf, month_plot, quarter_
plot
from sklearn.metrics import mean_squared_error, r2_score
from sklearn import metrics
from sklearn.model_selection import train_test_split
from math import sqrt
from statsmodels.tsa.api import ExponentialSmoothing, SimpleExpSmoothing, Holt
import statsmodels.api as sm
from statsmodels.tsa.arima_model import ARIMA, ARMA
from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.tsa.ar_model import AR, ARResults
from statsmodels.tsa.stattools import acovf, acf, pacf, pacf_yw, pacf_ols
from pandas.plotting import lag_plot
from statsmodels.tools.eval_measures import mse, rmse, meanabs, aic, bic
import pmdarima
from pmdarima import auto_arima
from statsmodels.tsa.seasonal import seasonal_decompose
import seaborn as sns
from statsmodels.tsa.stattools import adfuller
from sklearn.linear_model import LinearRegression
import datetime
from statsmodels.tsa.holtwinters import SimpleExpSmoothing, ExponentialSmoothing
from sklearn.preprocessing import MinMaxScaler
from statsmodels.tsa.statespace.tools import diff
from datetime import datetime
from datetime import date
import pylab

from pylab import rcParams
rcParams['figure.figsize'] = 14,9
plt.rcParams.update({'font.size': 14.5})
import warnings
warnings.filterwarnings('ignore')
```

In [ ]:
```
def adf_test(series,LEN,title=''):
    """
    Pass in a time series and an optional title, returns an ADF report
    """

    if LEN>6:
        mlag=None
    else:
        mlag=3
    result = adfuller(series.dropna(),autolag='AIC',maxlag=mlag)
    labels = ['ADF test statistic','p-value','# lags used','# observations']
    out = pd.Series(result[0:4],index=labels)

    for key,val in result[4].items():
        out[f'critical value ({key})']=val

    return result[1]
```

Visualização da série temporal

In [ ]:
```
df=pd.read_excel('timeseries.xlsx',index_col='Data',parse_dates=True)
df.index.freq= 'MS'
df['VOL']=df['VOL']/1000
plt.plot(df.VOL,'o-');
plt.xlabel('Data')
plt.ylabel('Volume fornecido [dam³]')
plt.autoscale(axis='x',tight=True)
```

In [ ]:
```
fig, axarr = plt.subplots(2, sharex=True)
fig.set_size_inches(10.5,8.5)

plot_pacf(df.VOL, ax=axarr[1],lags=30)
axarr[1].set_title('PACF')
plot_acf(df.VOL, ax=axarr[0],lags=30)
axarr[0].set_title('ACF')
```

Train-Test Split

In [ ]:
```
LEN=12
train_data=df[:-LEN]
test_data=df[-LEN:]
Fit=test_data[['VOL']].copy()
test_mean=test_data.VOL.mean()
```

In [ ]:
```
plt.plot(df.VOL,'bo-', label='Treino');
plt.plot(test_data.VOL,'co-', label='Teste');
plt.xlabel('Data')
plt.ylabel('Volume fornecido [dam³]')
pylab.legend(loc='upper left')
plt.autoscale(axis='x',tight=True)
```

Modelos Estocásticos

In [ ]:
```
LEN=12
fitted_model = ExponentialSmoothing(train_data['VOL'],trend='add', seasonal='mul',s
easonal_periods=12).fit()
Fit['Holt Winters']=fitted_model.forecast(LEN)
fitted_model.params
```

In [ ]:
```
auto_arima(train_data.VOL,m=12,maxiter=10).summary() #cálculo do AIC
```

In [ ]:
```
model = SARIMAX(train_data.VOL, order=(2,1,0),seasonal_order=(1,0,1,12))
SARIMA=model.fit()

SARIMA.summary()
```

In [ ]:
```
plt.plot(train_data.VOL[1:],'bo-',label='Treino')
plt.plot(SARIMA.fittedvalues[1:],'rx--',label='SARIMA(1,1,2)(1,0,1,12)')#xli
m=['2011-02-01','2018-03-01'],ylim=[150,350])
plt.plot(fitted_model.fittedvalues[1:],'g.--',label='Holt Winters')
plt.ylabel('Volume fornecido [dam³]');
plt.xlabel('Data');
plt.autoscale(axis='x',tight=True)
pylab.legend(loc='upper left')
```

In [ ]:
```
start=len(train_data)
end=len(train_data) + len(test_data) - 1

Fit['Holt-Winters']=fitted_model.forecast(LEN)
Fit['SARIMA']=SARIMA.predict(start,end,type='levels')

conca=train_data['VOL'][-1:]
frameHW=[conca,Fit['Holt-Winters']]
frameSARIMA=[conca,Fit['SARIMA']]
frametest=[conca,test_data.VOL]
HW=pd.concat(frameHW)
sarima=pd.concat(frameSARIMA)
test=pd.concat(frametest)
```

In [ ]:
```
rH=rmse(test_data.VOL,Fit['Holt-Winters'])
rS=rmse(test_data.VOL,Fit['SARIMA']);
print(rH,rS)
```

In [ ]:
```
nS=test_data.VOL-Fit['SARIMA']
nH=test_data.VOL-Fit['Holt-Winters']
print(adf_test(nS,LEN),adf_test(nH,LEN))
```

In [ ]:
```
plt.plot(train_data.VOL[70:],'bo-', label='Treino')
plt.plot(test_data.VOL,'co-', label='Teste')
plt.plot(sarima,'rx--',label='SARIMA(1,1,2)(1,0,1,12)')
plt.plot(HW,'g.--', label='Holt-Winters');
plt.autoscale(axis='x',tight=True)
pylab.legend(loc='down left')
plt.ylabel('Volume fornecido [dam³]');
plt.xlabel('Data');
```

In [ ]:
```
print('Médias: \nTeste:',test_mean,'\nHolt-Winters:',Fit['Holt-Winters'].mean(),'\n
SARIMA:',Fit['SARIMA'].mean())
print('\nDesvios padrão:\nHolt-Winters:',sqrt(nH.var()),'\nSARIMA:',sqrt(nS.var()))
```

ANÁLISE COMPORTAMENTO DINÂMICO

In [ ]:
```
varTr=df.VOL[48:]
plt.plot(df.VOL[:49],'bo-', label='Seção mínima de teste')
plt.plot(varTr,'co-', label='Seção de variação de treino')
plt.ylabel('Volume fornecido [dam³]');
plt.xlabel('Data');
plt.autoscale(axis='x',tight=True)
pylab.legend(loc='down center');
```

In [ ]:
```
analysis=pd.DataFrame(index=range(0),columns=['Model','Length of predictions','Last
train point',
                                              'RMSE','R²','Mean','Res Estacionário
','Test mean',
                                              'a','c','d','ar1','ar2','AR1','MA1'])
```

In [ ]:
```
vector=[12]
train_size=100

for LEN in vector:
    df1=df

    while train_size>48 :

        train_data=df1.iloc[:-LEN]
        test_data=df1.iloc[-LEN:]
        Fit=test_data[['VOL']].copy()

        test_mean=test_data.VOL.mean()

# 1) HOLT WINTERS

### MÉTODO DE HOLT WINTERS

        fitted_model = ExponentialSmoothing(train_data['VOL'],trend='add', seasona
l='mul',seasonal_periods=12).fit()
        Fit['Holt Winters']=fitted_model.forecast(LEN)

        Res_HW=test_data.VOL -Fit['Holt Winters']

        H=adf_test(Res_HW,LEN)
        if H<=0.05:
            end_estacionary='True'
        else:
            end_estacionary='False'


        analysis.loc['Holt Winters, '+' Len: ' + str(LEN) + ', trStart: '+
                     train_data.index[len(train_data)-1].strftime("%m/20%y")] = pd.
Series({
                                                    'Model': 'Holt Winters',
                                                    'Length of predictions':LE
N,
                                                    'Last train point':train_da
ta.index[len(train_data)-1],
                                                    'RMSE':rmeq,'R²': r2, 'Mean
':med,
                                                    'Res Estacionário': end_est
acionary,
                                                    'Test mean':test_mean,'a':
a,'c':c,'d':d})

        #datetime.date(train_data.index[len(train_data)-1]).strftime("%m/20%y")
# SARIMA

        model = SARIMAX(train_data.VOL, order=(2,1,0),seasonal_order=(1,0,1,12))
        SARIMA=model.fit()
          model = SARIMAX(train_data.VOL, order=(2,1,0),seasonal_orde
r=(1,0,1,12))
          SARIMA=model.fit()
          start=len(train_data)
          end=len(train_data) + len(test_data) - 1

          Fit['SARIMA']=SARIMA.predict(start,end,type='levels')

          Res_SARIMA=test_data.VOL -Fit['SARIMA']
          ar1=(-1)*model.polynomial_ar[1]
          ar2=(-1)*model.polynomial_ar[2]
          AR1=(-1)*model.polynomial_seasonal_ar[LEN]
          MA1=model.polynomial_seasonal_ma[LEN]


          HS=adf_test(Res_SARIMA,LEN)
          if HS<=0.05 and HS>0:
              end_estacionaryS='True'
          else:
              end_estacionaryS='False'

        rmeqS=rmse(test_data.VOL,Fit['SARIMA'])
          r2S=r2_score(test_data.VOL,Fit['SARIMA'])
          medS=Fit['SARIMA'].mean()
```

```
        analysis.loc['SARIMA, '+' Len: ' + str(LEN) + ', trStart: '+
                     train_data.index[len(train_data)-1].strftime("%m
/20%y")] = pd.Series({
                                                          'Model': 'SARIM
A',
                                                          'Length of pred
ictions':LEN,
                                                          'Last train poi
nt':train_data.index[len(train_data)-1],
                                                          'RMSE':rmeqS,'
R²': r2S, 'Mean':medS,
                                                          'Res Estacionár
io': end_estacionaryS,
                                                          'Test mean':tes
t_mean, 'ar1': ar1, 'ar2':ar2,
                                                        'AR1':AR1, 'MA1':
MA1})

        df1=df1[:-1]
        train_size=len(train_data)
```

In [ ]:
```
analysis['RMSE/Test mean']=(analysis['RMSE']/analysis['Test mean'])*100
```

In [ ]:
```
analysis.index=analysis['Last train point']
```

In [ ]:
```
analysis_12_SARIMA=analysis[(analysis['Model']=='SARIMA')]
analysis_12_HW=analysis[(analysis['Model']=='Holt Winters')]
```

In [ ]:
```
analysis.to_excel(r'Analysis.xlsx')
```

In [ ]:
```
### Criando a série de dado de média de todos pontos anteriores
Err_S=analysis_12_SARIMA['RMSE/Test mean']*0
Err_HW=analysis_12_HW['RMSE/Test mean']*0

Err_S[0]=analysis_12_SARIMA['RMSE/Test mean'][:(len(analysis_12_SARIMA)-1)].mean()
Err_HW[0]=analysis_12_HW['RMSE/Test mean'][:(len(analysis_12_HW)-1)].mean()

for i in range(1,(len(analysis_12_HW))):
    Err_S[i]=analysis_12_SARIMA['RMSE/Test mean'][i:].mean()
    Err_HW[i]=analysis_12_HW['RMSE/Test mean'][i:].mean()
```

In [ ]:
```
fig, ax1 = plt.subplots()

ax1.plot(analysis_12_HW['RMSE/Test mean'],'go-', label='Holt-Winters')
ax1.plot(Err_HW,'g--',label='RMEQ(%) médio Holt-Winters')

ax1.plot(analysis_12_SARIMA['RMSE/Test mean'],'ro-', label='SARIMA(2,1,0)
(1,0,1,12)')
ax1.plot(Err_S,'r--',label='RMEQ(%) médio SARIMA(2,1,0)(1,0,1,12)');

ax1.set_xlabel('Data de início da previsão')
ax1.set_ylabel('REQM(%)')

for x in analysis_12_SARIMA[(analysis_12_SARIMA['Res Estacionário']=='True')].inde
x:
    ax1.axvline(x=x, linestyle=':',color='k', label='Resíduo Estacionário (ambo
s)');
    print(x)

pylab.legend(loc='upper center');
ax2 = ax1.twinx()
ax2.plot(varTr[1:-LEN],'c',alpha=0.25,label='VOL')
ax2.set_ylabel('Volume fornecido [dam³]');

plt.autoscale(axis='x',tight=True)

pylab.legend(loc='down center');
```

In [ ]:
```
from pylab import *
subplot(3,1,1)
title('Coeficientes SARIMA(2,1,0)(1,0,1,12)')
plot(analysis_12_SARIMA.index[:-2],analysis_12_SARIMA['ar1'][:-2],'g.--',label='AR
(1)')
plot(analysis_12_SARIMA.index[:-2],analysis_12_SARIMA['ar2'][:-2],'r.--',label='AR
(2)')
pylab.legend(loc='upper right');
xticks([])

plt.autoscale(axis='x',tight=True)
subplot(3,1,2)
plot(analysis_12_SARIMA.index[:-2],analysis_12_SARIMA['AR1'][:-2],'b.--',label='AR
(1$^{12}$)')
pylab.legend(loc='lower right');
plt.autoscale(axis='x',tight=True)
xticks([])
plt.autoscale(axis='x',tight=True)
subplot(3,1,3)
plot(analysis_12_SARIMA.index[:-2],(analysis_12_SARIMA['MA1'])[:-2],'k.--',label='M
A(1$^{12}$)')
pylab.legend(loc='upper right');
plt.autoscale(axis='x',tight=True)
xlabel('Data de início de previsão')
```

In [ ]:
```
#analysis_12_HW[['a','b','c']]
subplot(3,1,1)
title('Coeficientes Holt-Winters')
plt.plot(analysis_12_HW.index,analysis_12_HW['a'],'g.--',label='a')
pylab.legend(loc='lower right');
xticks([])

plt.autoscale(axis='x',tight=True)
subplot(3,1,2)
plt.semilogy(analysis_12_HW.index,analysis_12_HW['b'],'b.--',label='c')
pylab.legend(loc='lower right');
plt.autoscale(axis='x',tight=True)
xticks([])

plt.autoscale(axis='x',tight=True)
subplot(3,1,3)
plt.semilogy(analysis_12_HW.index,analysis_12_HW['c'],'k.--',label='d')
pylab.legend(loc='upper center');
plt.autoscale(axis='x',tight=True)
plt.xlabel('Data de início de previsão')
```

Redes Neurais

In [ ]:
```
import keras
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from statsmodels.tsa.api import ExponentialSmoothing, SimpleExpSmoothing, Holt
from sklearn.preprocessing import MinMaxScaler
from keras.preprocessing.sequence import TimeseriesGenerator
```

In [ ]:
```
df=pd.read_excel('timeseries.xlsx',index_col='Data',parse_dates=True)
df.index.freq= 'MS'
df['VOL']=df['VOL']/1000
columns_to_keep = ['VOL']
df = df[columns_to_keep]
df.index.names = ['Data']
df.sort_index(inplace=True)
print('Total rows: {}'.format(len(df)))
```

In [ ]:
```
df=pd.read_excel('timeseries.xlsx',index_col='Data',parse_dates=True)
df.index.freq= 'MS'
df['VOL']=df['VOL']/1000
columns_to_keep = ['VOL']
df = df[columns_to_keep]
df.index.names = ['Data']
df.sort_index(inplace=True)
print('Total rows: {}'.format(len(df)))
```

In [ ]:
```
LEN=12
train = df.iloc[:-LEN]
test = df.iloc[-LEN:]
print('train: {}\ntest: {}'.format(len(train), len(test)))
```

In [ ]:
```
scaler = MinMaxScaler()

scaler.fit(train)

scaled_train = scaler.transform(train)
scaled_test = scaler.transform(test)
```

In [ ]:
```
n_input = 24
n_features = 1
Neu=35
generator = TimeseriesGenerator(scaled_train, scaled_train, length=n_input, batch_s
ize=1)
```

In [ ]:
```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
```

In [ ]:
```
model = Sequential()
model.add(LSTM(Neu, input_shape=(n_input, n_features)))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mean_squared_error')
```

In [ ]:
```
model.summary()
```

In [ ]:
```
ep=300
model.fit_generator(generator,epochs=ep);
```

In [ ]:
```
feval_b = scaled_train[-n_input:]
```

In [ ]:
```
feval_b = feval_b.reshape((1, n_input, n_features))
```

In [ ]:
```
model.predict(feval_b)
```

In [ ]:
```
test_predictions = []

feval_b = scaled_train[-n_input:]
c_b = feval_b.reshape((1, n_input, n_features))
```

In [ ]:
```
for i in range(len(test)):

    c_pred = model.predict(c_b)[0]

    test_predictions.append(c_pred)

    c_b = np.append(c_b[:,1:,:],[[c_pred]],axis=1)
```

In [ ]:
```
predictions = scaler.inverse_transform(test_predictions)
test['Predictions']=predictions

conca=train['VOL'][-1:]
frameNN=[conca,test['Predictions']]
frametest=[conca,test.VOL]
NN=pd.concat(frameNN)
testes=pd.concat(frametest)
lab='LSTM ('+str(Neu)+','+str(n_input)+','+str(ep)+')'

plt.plot(train.VOL[66:],'b', label='Treino')
plt.plot(testes,'c', label='Teste')
plt.plot(NN,'r--', label=lab);
plt.autoscale(axis='x',tight=True)
pylab.legend(loc='down left')
plt.ylabel('Volume fornecido [dam³]');
plt.xlabel('Data');
```

In [ ]:
```
from statsmodels.tools.eval_measures import mse, rmse
from sklearn.metrics import r2_score
re=rmse(test.VOL,test.Predictions)
r2=r2_score(test.VOL,test.Predictions)
```

In [ ]:
```
ADFT=adf_test((test.VOL-test.Predictions),LEN)

if ADFT<=0.05:
    b=True;
else:
    b=False;
```

APLICAÇÃO EM MÊS ANTERIOR

Mes escolhido: 07/2017

In [ ]:
```
print(analysis_12_SARIMA['RMSE/Test mean']['2017-07-01'])
```

In [ ]:
```
df17=df[:'2018-07-01']
train17=df17[:-LEN]
test17=df17[-LEN:]
scaler = MinMaxScaler()
scaled_train17=scaler.fit_transform(train17[['VOL']])
```

In [ ]:
```
n_input = 24
n_features = 1
Neu=35
generator = TimeseriesGenerator(scaled_train17, scaled_train17, length=n_input, bat
ch_size=1)

modelo = Sequential()
modelo.add(LSTM(Neu, input_shape=(n_input, n_features)))
modelo.add(Dense(1))
modelo.compile(optimizer='adam', loss='mean_squared_error')
```

In [ ]:
```
ep=300
modelo.fit_generator(generator,epochs=ep);
```

In [ ]:
```
feval_b = scaled_train17[-n_input:]

feval_b = feval_b.reshape((1, n_input, n_features))

modelo.predict(feval_b)

test_predictions = []

feval_b = scaled_train17[-n_input:]
c_b = feval_b.reshape((1, n_input, n_features))
```

In [ ]:
```
for i in range(len(test17)):
    #Prever 1 ponto
    c_pred = modelo.predict(c_b)[0]

    # Armazenar ponto
    test_predictions.append(c_pred)

    #Refazer batch eliminando o primeiro ponto e adicionando o previsto
    c_b = np.append(c_b[:,1:,:],[[c_pred]],axis=1)
```

In [ ]:
```
predictions = scaler.inverse_transform(test_predictions)
test17['LSTM(35,24,300)']=predictions

conca=train17['VOL'][-1:]
frameNN17=[conca,test17['LSTM(35,24,300)']]
frametest17=[conca,test17['VOL']]
NN17=pd.concat(frameNN17)
testes17=pd.concat(frametest17)
lab='LSTM('+str(Neu)+','+str(n_input)+','+str(ep)+')'

plt.plot(train17.VOL[25:],'bo-', label='Treino')
plt.plot(testes17,'co-', label='Teste')
plt.plot(NN17,'rx--', label=lab);
plt.autoscale(axis='x',tight=True)
pylab.legend(loc='down left')
plt.ylabel('Volume fornecido [dam³]');
plt.xlabel('Data');
```

In [ ]:
```
re17=rmse(test17.VOL,test17['LSTM(35,24,300)'])
r217=r2_score(test17.VOL,test17['LSTM(35,24,300)'])
print((re17),r217)
```

In [ ]:
```
ADFT=adf_test((test17.VOL-test17['LSTM(35,24,300)']),LEN)

if ADFT<=0.05:
    b=True;
else:
    b=False;
```

In [ ]:
```
model = SARIMAX(train17.VOL, order=(2,1,0),seasonal_order=(1,0,1,12))
res=model.fit()
start=len(train17)
end=len(train17) + len(test17) - 1

test17['SARIMA']=fitted_model.forecast(LEN)
test17['SARIMA']=res.predict(start,end,type='levels')

res=rmse(test17.VOL,test17['SARIMA'])
r2s=r2_score(test17.VOL,test17['SARIMA'])
```

In [ ]:
```
ADFT=adf_test((test17.VOL-test17['SARIMA']),LEN)

if ADFT<=0.05:
    b=True;
else:
    b=False;
```

In [ ]:
```
conca=train17['VOL'][-1:]
frameNN17=[conca,test17['LSTM(35,24,300)']]
frameSAR=[conca,test17['SARIMA']]
frametest17=[conca,test17['VOL']]
NN17=pd.concat(frameNN17)
SAR=pd.concat(frameSAR)
testes17=pd.concat(frametest17)
lab='LSTM('+str(Neu)+','+str(n_input)+','+str(ep)+')'

plt.plot(train17.VOL[25:],'bo-', label='Treino')
plt.plot(testes17,'co-', label='Teste')
plt.plot(NN17,'rx--', label=lab);
plt.plot(SAR,'g.--',label='SARIMA(2,1,0)(1,0,1,12)')
plt.autoscale(axis='x',tight=True)
pylab.legend(loc='down left')
plt.ylabel('Volume fornecido [dam³]');
plt.xlabel('Data');
```

VARIÁVEL EXÓGENA - DÓLAR

In [ ]:
```
df=pd.read_excel('timeseries.xlsx',index_col='Data',parse_dates=True)
df['VOL']=df['VOL']/1000
LEN=12
df1=df[:48]
treino = df1.iloc[:-LEN]
teste = df1.iloc[-LEN:]
testeplot=df1.iloc[(-LEN-1):]
fc=teste.copy()
```

In [ ]:
```
plt.plot(treino.VOL,'bo-',label='Treino')
plt.plot(testeplot.VOL,'co-',label='Teste')
plt.ylabel('Volume fornecido [dam³]');
plt.xlabel('Data');
plt.autoscale(axis='x',tight=True)
pylab.legend(loc='upper left')
```

In [ ]:
```
#auto_arima(treino.VOL,exogenous=treino[['Dolar (Compra)']],m=12).summary()
#auto_arima(treino.VOL,m=12).summary()
```

In [ ]:
```
model1 = SARIMAX(treino.VOL,exog=treino[['Dolar (Compra)']], order=(1,0,1), seasona
l_order=(2,0,0,12))
```

In [ ]:
```
sarimax1=model1.fit()
#sarimax1.summary()
```

In [ ]:
```
xtrain=treino[['Dolar (Compra)']]
ytrain=treino[['VOL']]
ln=LinearRegression()
ln.fit(xtrain,ytrain);

x=treino[['Dolar (Compra)']]
treino['Linear Regression']=ln.predict(x);

treino['Linear Residual']=treino.VOL/treino['Linear Regression']

fc['LR Forecast']=ln.predict(teste[['Dolar (Compra)']])
print(ln.coef_,ln.intercept_)
```

In [ ]:
```
auto_arima(treino['Linear Residual']).summary()
```

In [ ]:
```
model0=SARIMAX(treino.VOL, order=(2,1,0), seasonal_order=(1,0,0,12))
sarim=model0.fit()
sarim.summary()
```

In [ ]:
```
start=len(treino)
end=len(treino) + len(teste) - 1
fc['SARIMA']=sarim.predict(start,end,type='levels')
fc['SARIMAX']=sarimax1.predict(start,end,exog=teste[['Dolar (Compra)']],type='level
s')
```

In [ ]:
```
model2=SARIMAX(treino['Linear Residual'],order=(1,0,1))
arim=model2.fit()
#arim.summary()
```

In [ ]:
```
fc['Res ARIMA']=arim.predict(start,end,type='levels')
fc['LR + ARIMA']=fc['Res ARIMA']+fc['LR Forecast']
```

In [ ]:
```
conca=treino['VOL'][-1:]

Sframe=[conca,fc['SARIMA']]
Sfc=pd.concat(Sframe)
SXframe=[conca,fc['SARIMAX']]
SXfc=pd.concat(SXframe)
LAframe=[conca,fc['LR + ARIMA']]
LAfc=pd.concat(LAframe)

frametst=[conca,teste.VOL]
testes=pd.concat(frametst)

plt.plot(treino.VOL[25:],'bo-',label='Treino')
plt.plot(testes,'co-',label='Teste')
plt.plot(Sfc,'g*--',label='SARIMA(2,1,0)(1,0,0,12)')
plt.plot(SXfc,'ro--',label='SARIMAX(1,1,0)(2,0,0,12)')
plt.plot(LAfc,color='k',linestyle='--',marker='x',label='Regressão Linear + ARMA
(1,1)')
pylab.legend(loc='down left')
plt.autoscale(axis='x',tight=True)
plt.ylabel('Volume fornecido [dam³]');
plt.xlabel('Data');
```

In [ ]:
```
re=rmse(teste.VOL,fc['LR + ARIMA'])
r2=r2_score(teste.VOL,fc['LR + ARIMA'])

ADFT=adf_test((teste.VOL-fc['LR + ARIMA']),LEN)

if ADFT<=0.05:
    b=True;
else:
    b=False;

print(ADFT,b)

print(re,r2)
```

Códigos - TSA

In [ ]:
```
import keras
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from statsmodels.tsa.api import ExponentialSmoothing, SimpleExpSmoothing, Holt
from sklearn.preprocessing import MinMaxScaler
from keras.preprocessing.sequence import TimeseriesGenerator
from pylab import rcParams
rcParams['figure.figsize'] = 16,5
plt.rcParams.update({'font.size': 14.5})
import warnings
warnings.filterwarnings('ignore')
```

In [ ]:
```
df=pd.read_excel('delta P final.xlsx',index_col='t',parse_dates=True)
len(df)
```

In [ ]:
```
from statsmodels.tsa.stattools import adfuller
def adf_test(series,LEN,title=''):
    """
    Pass in a time series and an optional title, returns an ADF report
    """

    if LEN>6:
        mlag=None
    else:
        mlag=3
    result = adfuller(series.dropna(),autolag='AIC',maxlag=mlag) # .dropna() handle
s differenced data

    labels = ['ADF test statistic','p-value','# lags used','# observations']
    out = pd.Series(result[0:4],index=labels)

    for key,val in result[4].items():
        out[f'critical value ({key})']=val

    return result[1]
```

In [ ]:
```
train=df[:2536]
test=df[2536:]
```

In [ ]:
```
plt.plot(train,label='Treino',color='b')
plt.plot(test,label='Teste',color='c')
pylab.legend(loc='down left')
plt.ylabel('Diferença de pressão (Pa)');
plt.xlabel('Tempo (min)');
plt.autoscale(axis='x',tight=True)
```

In [ ]:
```
scaled_train = train.values
```

In [ ]:
```
n_input = 75
n_features = 1
Neu=50
generator = TimeseriesGenerator(scaled_train, scaled_train, length=n_input, batch_s
ize=1)
```

In [ ]:
```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
```

In [ ]:
```
model = Sequential()
model.add(LSTM(Neu, input_shape=(n_input, n_features)))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mean_squared_error')
```

In [ ]:
```
model.summary()
```

In [ ]:
```
ep=400
model.fit_generator(generator,epochs=ep);
```

In [ ]:
```
loss = model.history.history['loss']
plt.plot(range(len(loss)),loss);
plt.xlabel('Número de épocas')
plt.ylabel('EQM')
```

In [ ]:
```
feval_b = scaled_train[-n_input:]
feval_b = feval_b.reshape((1, n_input, n_features))
model.predict(feval_b)
test_predictions = []

feval_b = scaled_train[-n_input:]
c_b = feval_b.reshape((1, n_input, n_features))
```

In [ ]:
```
for i in range(len(test)):

    c_pred = model.predict(c_b)[0]

    test_predictions.append(c_pred)

    c_b = np.append(c_b[:,1:,:],[[c_pred]],axis=1)
```

In [ ]:
```
predictions = test_predictions
test['Predictions']=predictions
```

In [ ]:
```
import pylab

conca=train['dP'][-1:]
frameNN=[conca,test['Predictions']]
frametest=[conca,test.dP]
NN=pd.concat(frameNN)
testes=pd.concat(frametest)
lab='LSTM ('+str(Neu)+','+str(n_input)+','+str(ep)+')'

#plt.plot(train.dP,'b', label='Treino')
plt.plot(testes,'c', label='Teste')
plt.plot(NN,'k--', label=lab);
plt.autoscale(axis='x',tight=True)
pylab.legend(loc='lower right')
plt.ylabel('Volume fornecido [dam³]');
plt.xlabel('Data');
```

In [ ]:
```
from statsmodels.tools.eval_measures import mse, rmse
from sklearn.metrics import r2_score
re=rmse(test.dP,test['Predictions'])
r2=r2_score(test.dP,test['Predictions'])
print(re,r2)
```

In [ ]:
```
LEN=len(test)
```

In [ ]:
```
test['LSTM(50,50,300)']=test['Predictions']
REQM=re
R2=r2
```

In [ ]:
```
adf_test(test.dP - test['Predictions'],LEN)
```

Melhor resultado:

In [ ]:
```
conca=train['dP'][-1:]
frameNN=[conca,test['LSTM(50,50,300)']]
frametest=[conca,test.dP]
NN=pd.concat(frameNN)
testes=pd.concat(frametest)
lab='LSTM (50,50,300)'

#plt.plot(train.dP,'b', label='Treino')
plt.plot(testes,'c', label='Teste')
plt.plot(NN,'k--', label=lab);
plt.autoscale(axis='x',tight=True)
pylab.legend(loc='upper left')
plt.ylabel('Diferença de pressão [Pa]');
plt.xlabel('Tempo (10 x min)');
```

In [ ]:
```
C30=pd.read_excel('C30.xlsx',index_col='Ciclo',parse_dates=True)
C50=pd.read_excel('C50.xlsx',index_col='Ciclo',parse_dates=True)
C75=pd.read_excel('C75.xlsx',index_col='Ciclo',parse_dates=True)
print(len(C30),len(C50),len(C75))
```

In [ ]:
```
import pylab

plt.plot(C30.Resistencia,label='30% de adsorção')
plt.plot(C50.Resistencia,label='50% de adsorção')
plt.plot(C75.Resistencia,label='75% de adsorção')
plt.plot(lim.Resistencia,label='Limite',color='k',linestyle='--');
plt.ylabel('Resistência ao escoamento  ['+r'$\frac{h}{m^{3}} \sqrt{Pa}$]')
plt.xlabel('Ciclo')
pylab.legend(loc='down left')
plt.autoscale(axis='x',tight=True)
```

30%

In [ ]:
```
train_30=C30[:100]
test_30=C30[100:]
LEN=len(train_30)
```

In [ ]:
```
scaler = MinMaxScaler()

scaler.fit(train_30)
scaled_train_30=scaler.transform(train_30)
```

In [ ]:
```
n_input = 30
n_features = 1
Neu=100
generator = TimeseriesGenerator(scaled_train_30, scaled_train_30, length=n_input, b
atch_size=1)
```

In [ ]:
```
model = Sequential()
model.add(LSTM(Neu, input_shape=(n_input, n_features)))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mean_squared_error')
model.summary()
```

In [ ]:
```
ep=420
model.fit_generator(generator,epochs=20);
```

In [ ]:
```
loss = model.history.history['loss']
plt.plot(range(len(loss)),loss);
plt.xlabel('Número de épocas')
plt.ylabel('EQM')
```

In [ ]:
```
feval_b = scaled_train_30[-n_input:]
feval_b = feval_b.reshape((1, n_input, n_features))
model.predict(feval_b)
test_predictions = []

feval_b = scaled_train_30[-n_input:]
c_b = feval_b.reshape((1, n_input, n_features))
```

In [ ]:
```
for i in range(len(test_30)):

    c_pred = model.predict(c_b)[0]

    test_predictions.append(c_pred)

    c_b = np.append(c_b[:,1:,:],[[c_pred]],axis=1)
```

In [ ]:
```
predictions = scaler.inverse_transform(test_predictions)
test_30['Predictions']=predictions
```

In [ ]:
```
conca=train_30['Resistencia'][-1:]
frameNN=[conca,test_30['Predictions']]
frametest=[conca,test_30.Resistencia]
NN=pd.concat(frameNN)
testes=pd.concat(frametest)
lab='LSTM ('+str(Neu)+','+str(n_input)+','+str(ep)+')'

plt.plot(train_30.Resistencia,'b', label='Treino')
plt.plot(testes,'c', label='Teste')
plt.plot(NN,'k--', label=lab);
plt.autoscale(axis='x',tight=True)
pylab.legend(loc='down left')
plt.ylabel('Volume fornecido [dam³]');
plt.xlabel('Data');
```

In [ ]:
```
from statsmodels.tools.eval_measures import mse, rmse
from sklearn.metrics import r2_score
re=rmse(test_30.Resistencia,test_30['Predictions'])
r2=r2_score(test_30.Resistencia,test_30['Predictions'])
print(re,r2)
```

In [ ]:
```
model30=model
test_30[lab]=test_30.Predictions
```

In [ ]:
```
#analysis=pd.DataFrame(index=range(0),columns=['Model','RMSE','Mean'])
#ii=0
analysis.loc[ii] = pd.Series({
'Model': lab,'RMSE':re,'Mean':test_30[lab].mean()})
ii=ii+1
```

In [ ]:
```
conca=train_30['Resistencia'][-1:]
frameNN=[conca,test_30['LSTM (100,30,400)']]
frametest=[conca,test_30.Resistencia]
NN=pd.concat(frameNN)
testes=pd.concat(frametest)
lab='LSTM ('+str(Neu)+','+str(n_input)+','+str(ep)+')'

plt.plot(train_30.Resistencia,'b', label='Treino')
plt.plot(testes,'c', label='Teste')
plt.plot(NN,'k--', label=lab);
plt.autoscale(axis='x',tight=True)
pylab.legend(loc='lower right')
plt.ylabel('Resistência ao escoamento  ['+r'$\frac{h}{m^{3}} \sqrt{Pa}$]')
plt.xlabel('Ciclo');
```

50%

In [ ]:
```
train_50=C50[:100]
test_50=C50[100:]
LEN=len(train_50)
```

In [ ]:
```
scaler = MinMaxScaler()

scaler.fit(train_50)
scaled_train_50=scaler.transform(train_50)
```

In [ ]:
```
n_input = 30
n_features = 1
Neu=100
generator = TimeseriesGenerator(scaled_train_50, scaled_train_50, length=n_input, b
atch_size=1)
```

In [ ]:
```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
```

In [ ]:
```
model = Sequential()
model.add(LSTM(Neu, input_shape=(n_input, n_features)))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mean_squared_error')
model.summary()
```

In [ ]:
```
ep=400
model.fit_generator(generator,epochs=400);
```

In [ ]:
```
loss = model.history.history['loss']
plt.plot(range(len(loss)),loss);
plt.xlabel('Número de épocas')
plt.ylabel('EQM')
```

In [ ]:
```
feval_b = scaled_train_50[-n_input:]
feval_b = feval_b.reshape((1, n_input, n_features))
model.predict(feval_b)
test_predictions = []

feval_b = scaled_train_50[-n_input:]
c_b = feval_b.reshape((1, n_input, n_features))
```

In [ ]:
```
for i in range(len(test_50)):

    c_pred = model.predict(c_b)[0]

    test_predictions.append(c_pred)

    c_b = np.append(c_b[:,1:,:],[[c_pred]],axis=1)
```

In [ ]:
```
predictions = scaler.inverse_transform(test_predictions)
test_50['Predictions']=predictions
```

In [ ]:
```
conca=train_50['Resistencia'][-1:]
frameNN=[conca,test_50['Predictions']]
frametest=[conca,test_50.Resistencia]
NN=pd.concat(frameNN)
testes=pd.concat(frametest)
lab='LSTM ('+str(Neu)+','+str(n_input)+','+str(ep)+')'

plt.plot(train_50.Resistencia,'b', label='Treino')
plt.plot(testes,'c', label='Teste')
plt.plot(NN,'k--', label=lab);
plt.autoscale(axis='x',tight=True)
pylab.legend(loc='down left')
plt.ylabel('Volume fornecido [dam³]');
plt.xlabel('Data');
```

In [ ]:
```
from statsmodels.tools.eval_measures import mse, rmse
from sklearn.metrics import r2_score
re=rmse(test_50.Resistencia,test_50['Predictions'])
r2=r2_score(test_50.Resistencia,test_50['Predictions'])
print(re,r2)
```

In [ ]:
```
model50=model
test_50[lab]=test_50.Predictions
```

In [ ]:
```
an50=pd.DataFrame(index=range(0),columns=['Model','RMSE','Mean'])
#ii=0
an50.loc[ii] = pd.Series({
'Model': lab,'RMSE':re,'Mean':test_50[lab].mean()})
ii=ii+1
```

In [ ]:
```
ADFT=adf_test((test_50['Resistencia']-test_50['LSTM (100,30,400)']),LEN)

if ADFT<=0.05:
    b=True;
else:
    b=False;

print(ADFT,b)
```

In [ ]:
```
conca=train_50['Resistencia'][-1:]
frameNN=[conca,test_50['LSTM (100,30,400)']]
frametest=[conca,test_50.Resistencia]
NN=pd.concat(frameNN)
testes=pd.concat(frametest)
lab='LSTM ('+str(Neu)+','+str(n_input)+','+str(ep)+')'

plt.plot(train_50.Resistencia,'b', label='Treino')
plt.plot(testes,'c', label='Teste')
plt.plot(NN,'k--', label=lab);
plt.autoscale(axis='x',tight=True)
pylab.legend(loc='lower right')
plt.ylabel('Resistência ao escoamento  ['+r'$\frac{h}{m^{3}} \sqrt{Pa}$]')
plt.xlabel('Ciclo');
```

75%

In [ ]:
```
train_75=C75[:100]
test_75=C75[100:]
LEN=len(train_75)
```

In [ ]:
```
scaler = MinMaxScaler()

scaler.fit(train_75)
scaled_train_75=scaler.transform(train_75)
```

In [ ]:
```
n_input = 30
n_features = 1
Neu=100
generator = TimeseriesGenerator(scaled_train_75, scaled_train_75, length=n_input, b
atch_size=1)
```

In [ ]:
```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
```

In [ ]:
```
model = Sequential()
model.add(LSTM(Neu, input_shape=(n_input, n_features)))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mean_squared_error')
model.summary()
```

In [ ]:
```
ep=400
model.fit_generator(generator,epochs=ep);
```

In [ ]:
```
loss = model.history.history['loss']
plt.plot(range(len(loss)),loss);
plt.xlabel('Número de épocas')
plt.ylabel('EQM')
```

In [ ]:
```
feval_b = scaled_train_75[-n_input:]
feval_b = feval_b.reshape((1, n_input, n_features))
model.predict(feval_b)
test_predictions = []

feval_b = scaled_train_75[-n_input:]
c_b = feval_b.reshape((1, n_input, n_features))
```

In [ ]:
```
for i in range(len(test_75)):

    c_pred = model.predict(c_b)[0]

    test_predictions.append(c_pred)

    c_b = np.append(c_b[:,1:,:],[[c_pred]],axis=1)
```

In [ ]:
```
predictions = scaler.inverse_transform(test_predictions)
test_75['Predictions']=predictions
```

In [ ]:
```
conca=train_75['Resistencia'][-1:]
frameNN=[conca,test_75['Predictions']]
frametest=[conca,test_75.Resistencia]
NN=pd.concat(frameNN)
testes=pd.concat(frametest)
lab='LSTM ('+str(Neu)+','+str(n_input)+','+str(ep)+')'

plt.plot(train_75.Resistencia,'b', label='Treino')
plt.plot(testes,'c', label='Teste')
plt.plot(NN,'k--', label=lab);
plt.autoscale(axis='x',tight=True)
pylab.legend(loc='down left')
plt.ylabel('Volume fornecido [dam³]');
plt.xlabel('Data');
```

In [ ]:
```
from statsmodels.tools.eval_measures import mse, rmse
from sklearn.metrics import r2_score
re=rmse(test_75.Resistencia,test_75['Predictions'])
r2=r2_score(test_75.Resistencia,test_75['Predictions'])
print(re,r2)
```

In [ ]:
```
conca=train_75['Resistencia'][-1:]
frameNN=[conca,test_75['LSTM (100,30,400)']]
frametest=[conca,test_75.Resistencia]
NN=pd.concat(frameNN)
testes=pd.concat(frametest)
lab='LSTM ('+str(Neu)+','+str(n_input)+','+str(ep)+')'

plt.plot(train_75.Resistencia,'b', label='Treino')
plt.plot(testes,'c', label='Teste')
plt.plot(NN,'k--', label=lab);
plt.autoscale(axis='x',tight=True)
pylab.legend(loc='lower right')
plt.ylabel('Resistência ao escoamento  ['+r'$\frac{h}{m^{3}} \sqrt{Pa}$]')
plt.xlabel('Ciclo');
```

In [ ]:
```
df=pd.read_excel('Ciclos FC.xlsx',index_col='Ciclo',parse_dates=True)
C30=pd.read_excel('C30.xlsx',index_col='Ciclo',parse_dates=True)
C50=pd.read_excel('C50.xlsx',index_col='Ciclo',parse_dates=True)
C75=pd.read_excel('C75.xlsx',index_col='Ciclo',parse_dates=True)
```

In [ ]:
```
lim=C50*0
for j in range(1,161):
    lim.loc[j]=8
```

In [ ]:
```
C30['LSTM 100,30,400']=df['C30 (100,30,400)']
C30['LSTM 100,36,450']=df['C30(100,36,450)']
C50['LSTM 100,30,400']=df['C50 (100,30,400)']
C50['LSTM 100,36,450']=df['C50(100,36,450)']
C75['LSTM 100,30,400']=df['C75 (100,30,400)']
C75['LSTM 100,36,450']=df['C75(100,36,450)']
```

In [ ]:
```
test30=C30[101:]
train30=C30[:101]
test50=C50[101:]
train50=C50[:101]
test75=C75[101:]
train75=C75[:101]
```

30% Ciclo

In [ ]:
```
conca=train30['Resistencia'][-1:]
frameNN1=[conca,test30['LSTM 100,30,400']]
frameNN2=[conca,test30['LSTM 100,36,450']]
frametest=[conca,test30.Resistencia]
NN1=pd.concat(frameNN1)
NN2=pd.concat(frameNN2)
testes=pd.concat(frametest)

plt.plot(train30.Resistencia,'b', label='Treino')
plt.plot(testes,'c', label='Teste')
plt.plot(NN1,'r--', label='LSTM (100,30,400)');
plt.plot(NN2,'g--', label='LSTM (100,36,450)');
#plt.plot(lim.Resistencia,label='Limite',color='k',linestyle='--');
plt.autoscale(axis='x',tight=True)
pylab.legend(loc='lower right')
plt.ylabel('Resistência ao escoamento  ['+r'$\frac{h}{m^{3}} \sqrt{Pa}$]')
plt.xlabel('Ciclo');
```

In [ ]:
```
rel=rmse(test30.Resistencia,test30['LSTM 100,30,400'])
r2_1=r2_score(test30.Resistencia,test30['LSTM 100,30,400'])
ADF1=adf_test((test30['Resistencia']-test30['LSTM 100,30,400']),len(test30))

re2=rmse(test30.Resistencia,test30['LSTM 100,36,450'])
r2_2=r2_score(test30.Resistencia,test30['LSTM 100,36,450'])
ADF2=adf_test((test30['Resistencia']-test30['LSTM 100,36,450']),len(test30))


print('30% do ciclo de adsorção: RMSE          R²                      p\nLSTM
100,30,400: ',rel,r2_1,ADF1,'\nLSTM 100,36,450: ',re2,r2_2,ADF2)
```

50% Ciclo

In [ ]:
```
conca=train50['Resistencia'][-1:]
frameNN1=[conca,test50['LSTM 100,30,400']]
frameNN2=[conca,test50['LSTM 100,36,450']]
frametest=[conca,test50.Resistencia]
NN1=pd.concat(frameNN1)
NN2=pd.concat(frameNN2)
testes=pd.concat(frametest)

plt.plot(train50.Resistencia,'b', label='Treino')
plt.plot(testes,'c', label='Teste')
plt.plot(NN1,'r--', label='LSTM (100,30,400)');
plt.plot(NN2,'g--', label='LSTM (100,36,450)');
#plt.plot(lim.Resistencia,label='Limite',color='k',linestyle='--');
plt.autoscale(axis='x',tight=True)
pylab.legend(loc='lower right')
plt.ylabel('Resistência ao escoamento  ['+r'$\frac{h}{m^{3}} \sqrt{Pa}$]')
plt.xlabel('Ciclo');
```

In [ ]:
```
rel=rmse(test50.Resistencia,test50['LSTM 100,30,400'])
r2_1=r2_score(test50.Resistencia,test50['LSTM 100,30,400'])
ADF1=adf_test((test50['Resistencia']-test50['LSTM 100,30,400']),len(test30))

re2=rmse(test50.Resistencia,test50['LSTM 100,36,450'])
r2_2=r2_score(test50.Resistencia,test50['LSTM 100,36,450'])
ADF2=adf_test((test50['Resistencia']-test50['LSTM 100,36,450']),len(test30))


print('50% do ciclo de adsorção: RMSE          R²                      p\nLSTM
100,30,400: ',rel,r2_1,ADF1,'\nLSTM 100,36,450: ',re2,r2_2,ADF2)
```

75% Ciclo

In [ ]:
```
conca=train75['Resistencia'][-1:]
frameNN1=[conca,test75['LSTM 100,30,400']]
frameNN2=[conca,test75['LSTM 100,36,450']]
frametest=[conca,test75.Resistencia]
NN1=pd.concat(frameNN1)
NN2=pd.concat(frameNN2)
testes=pd.concat(frametest)

plt.plot(train75.Resistencia,'b', label='Treino')
plt.plot(testes,'c', label='Teste')
plt.plot(NN1,'r--', label='LSTM (100,30,400)');
plt.plot(NN2,'g--', label='LSTM (100,36,450)');
#plt.plot(lim.Resistencia,label='Limite',color='k',linestyle='--');
plt.autoscale(axis='x',tight=True)
pylab.legend(loc='lower right')
plt.ylabel('Resistência ao escoamento  ['+r'$\frac{h}{m^{3}} \sqrt{Pa}$]')
plt.xlabel('Ciclo');
```

In [ ]:
```
rel=rmse(test75.Resistencia,test75['LSTM 100,30,400'])
r2_1=r2_score(test75.Resistencia,test75['LSTM 100,30,400'])
ADF1=adf_test((test75['Resistencia']-test75['LSTM 100,30,400']),len(test30))

re2=rmse(test75.Resistencia,test75['LSTM 100,36,450'])
r2_2=r2_score(test75.Resistencia,test75['LSTM 100,36,450'])
ADF2=adf_test((test75['Resistencia']-test75['LSTM 100,36,450']),len(test30))


print('75% do ciclo de adsorção: RMSE          R²                      p\nLSTM
100,30,400: ',rel,r2_1,ADF1,'\nLSTM 100,36,450: ',re2,r2_2,ADF2)
```