DUAS PROPOSTAS PARA SOLUCIONAR O PROBLEMA REFERENTE A MÉDIAS MÓVEIS NULAS NA NORMALIZAÇÃO ADAPTATIVA

Eduardo Fernando Costa de Niemeyer

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Geraldo Zimbrão da Silva

Rio de Janeiro
Setembro de 2018

DUAS PROPOSTAS PARA SOLUCIONAR O PROBLEMA REFERENTE A MÉDIAS MÓVEIS NULAS NA NORMALIZAÇÃO ADAPTATIVA

Eduardo Fernando Costa de Niemeyer

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Geraldo Zimbrão da Silva, D.Sc.

Prof. Eduardo Soares Ogasawara, D.Sc.

Prof. Geraldo Bonorino Xexéo, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2018

Niemeyer, Eduardo Fernando Costa de

Duas propostas para solucionar o problema referente a médias móveis nulas na Normalização Adaptativa/Eduardo Fernando Costa de Niemeyer. – Rio de Janeiro: UFRJ/COPPE, 2018.

XII, 76 p.: il.; 29, 7cm.

Orientador: Geraldo Zimbrão da Silva

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2018.

Referências Bibliográficas: p. 71 – 76.

1. Normalização. 2. Séries Temporais. 3. Redes Neurais. I. Silva, Geraldo Zimbrão da. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

À minha família.

Agradecimentos

Realizar pesquisa acadêmica não é uma tarefa fácil, há muitos desafios, desânimo, ansiedade, medo, desconfiança, entre outros. Se consegui concluir essa dissertação devo e muito a algumas pessoas que me ajudaram e não me deixaram desistir.

Agradeço, primeiramente, ao meu orientador, prof. Geraldo Zimbrão, por se interessar pelo meu tema e me indicar os caminhos a seguir, sendo um dos caminhos sugeridos conversar com Eduardo Ogasawara e me aproximar do tema Normalização Adaptativa. Também sou muito grato ao Eduardo Ogasawara, que me tirou muitas dúvidas e deu dicas importantíssimas, praticamente me colocando nos trilhos para terminar a dissertação.

Sou grato pelos companheiros que fiz no PESC, assim como a equipe de trabalho da qual eu faço parte na Fundação Coppetec, que também me ajudaram com dicas sobre pesquisa; apontaram erros; alertaram sobre prazos; ajudaram com dicas de tecnologia a utilizar (o uso da biblioteca Keras foi uma delas); e também pelo seu apoio e por sempre acreditarem na minha capacidade.

Por último, devo agradecer ao suporte familiar, sem o amor e o investimento deles na minha educação eu não teria chegado ao ponto que cheguei. Dedico esse trabalho principalmente a eles.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

DUAS PROPOSTAS PARA SOLUCIONAR O PROBLEMA REFERENTE A MÉDIAS MÓVEIS NULAS NA NORMALIZAÇÃO ADAPTATIVA

Eduardo Fernando Costa de Niemeyer

Setembro/2018

Orientador: Geraldo Zimbrão da Silva

Programa: Engenharia de Sistemas e Computação

Apresenta-se, nesta dissertação, duas novas propostas para sanar o problema das médias móveis nulas para a Normalização Adaptativa, utilizada como parte do pré-processamento de um método de aprendizagem de máquinas para previsão em séries temporais. São elas: Normalização Adaptativa Compensada e Normalização Adaptativa por Subtração.

Compara-se e analisa-se os resultados obtidos com essas duas novas propostas, aplicadas a 5 diferentes *Datasets*, com a Normalização Adaptativa Original, assim como com outros métodos de normalização presentes na literatura e o *baseline* ARIMA. Conclui-se que, a Normalização Adaptativa por Subtração, proposta nesse trabalho, supera todos os outros métodos para séries temporais não-estacionárias e heteroscedásticas, também corrigindo o problema referente às médias móveis nulas para a Normalização Adaptativa Original. Para séries temporais com alto grau de estacionariedade, todos os métodos de Normalização Adaptativa não são satisfatórios.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

TWO PROPOSALS TO SOLVE THE PROBLEM CONCERNING NULL MOVING AVERAGES IN ADAPTIVE NORMALIZATION

Eduardo Fernando Costa de Niemeyer

September/2018

Advisor: Geraldo Zimbrão da Silva

Department: Systems Engineering and Computer Science

This dissertation presents two new proposals to solve the problem of null moving averages for Adaptive Normalization, used as part of the pre-processing phase of a machine learning method for time series forecasting. They are: Adaptive Normalization by Compensation and Adaptive Normalization by Subtraction.

It compares and analyzes the results obtained with these two new proposals, applied to 5 different *datasets*, with the Original Adaptive Normalization, as well as with other normalization methods present in the literature and the *baseline* ARIMA. It is concluded that, the Adaptive Normalization by Subtraction, proposed in this work, surpasses all other methods when it is applied to non-stationary heteroscedastic time series, also correcting the problem related to null moving averages of the Original Adaptive Normalization. For time series with high degree of stationarity, all the Adaptive Normalization methods are not satisfatory.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Motivação e Contexto

Há tempos, muitos estudiosos, profissionais do ramo financeiro e investidores buscam encontrar meios de lucrar com investimentos. Se antigamente era possível lucrar através de comércio, comprando um produto a um preço baixo e o revendendo a um preço maior, hoje já é possível realizar o mesmo tipo de negociação de forma mais simples através do mercado financeiro, assim como em outras aplicações como opções, contratos e minicontratos futuros, derivativos, câmbio, entre outros e, mais recentemente, criptomoedas também estão sendo utilizadas para realizar *trading*.

O mercado de capitais fora concebido com a ideia de permitir que se aplique capital em empresas, obtendo-se papéis que representam um pequeno percentual de participação nela. Desta forma, a empresa obtém recursos do investidor, que, por sua vez, ganha parte dos lucros da empresa e com o seu crescimento, devido a valorização dos papéis. Porém, os preços de ações nem sempre refletem o valor da empresa de fato e ocorrem flutuações devido a notícias, oferta e demanda, especulação, liquidez, entre outros fatores. *Traders* se utilizam dessas flutuações para tentar lucrar em um curto espaço de tempo.

Segundo a Hipótese do Mercado Eficiente (MALKIEL e FAMA, 1970), não é possível alcançar retornos consistentes superiores à média do mercado, já que o preço de um ativo representaria toda a informação histórica disponível até hoje, e qualquer lucro no curto prazo seria baseado puramente em sorte. Existem três vertentes dessa hipótese: a Fraca, Semiforte e Forte. A Fraca considera que toda informação pública está precificada no ativo. A Semiforte agrega à anterior que qualquer nova informação pública é precificada muito rapidamente, impedindo retornos excessivos. A Forte afirma que até informações não públicas, "privilegiadas", são refletidas quase instantaneamente nos preços de ativos. Essa teoria inviabilizaria a realização de *trading* de forma eficaz, onde se utiliza a análise técnica (análise de gráficos

com preços e volumes históricos para tentar prever variações futuras). Segundo a teoria, por simples questão de distribuição e probabilidades, seria possível existir alguns "vencedores" apenas por sorte, os quais conseguiriam lucros consistentes com essa prática, pois os preços de ativos seguiriam um *random walk* (FAMA, 1965), impossível de prever.

Segundo a análise fundamentalista, procura-se precificar um ativo baseado na análise da saúde financeira da empresa e perspectivas futuras, ou seja, os fundamentos da empresa, quantitativamente como lucro ou dívida, e qualitativamente como eficiência dos gestores, executivos e controladores. Dessa forma, pode-se supor que um determinado ativo está barato ou caro hoje e, no longo prazo, lucrar. Por ser uma previsão de longo prazo, evitam-se volatilidades, sendo mais seguro que o investimento de curto prazo, típico da análise técnica. Porém, supondo que se consiga encontrar um método eficiente de obter lucros no curto prazo, eles seriam muito maiores que os obtidos no longo prazo. Por isso, muitos analistas buscam encontrar padrões nos gráficos, criam métodos, regras, visando achar soluções que acertem mais nas previsões. Por que, então, não utilizar computadores para tentar prever? Hoje em dia, as máquinas realizam cálculos de forma muito mais precisa e rápida que os seres humanos, e com o desenvolvimento da Inteligência Artificial (IA), as previsões baseadas em dados históricos podem ser bem melhores do que as de muitos analistas técnicos experientes.

Não é por acaso, nem fato recente, que muitos pesquisadores da área de IA tem buscado aplicar seus conhecimentos para tentar prever os rumos do mercado acionário, pois, caso obtenham sucesso, as possibilidades de lucros são enormes. As soluções propostas variam da utilização de notícias e análise de sentimentos sobre elas, dados de fundamentos das empresas e suas concorrentes, até, claro, dados históricos, que incluem Preço de Abertura, Máximo, Mínimo, Fechamento e Volume para determinado período de tempo.

Notícias também são capazes de alterar os rumos do Mercado. Assim que é anunciado algo relevante, como algum desastre ambiental que prejudique uma colheita, a falência de uma empresa, ou até lucros acima do esperado, as ações das empresas prejudicadas ou beneficiadas pelo fato publicado sofrem rapidamente uma variação. Vale ressaltar que quando notícias são publicadas, normalmente os grandes *players* do mercado já estão sabendo antes, o que acaba sendo injusto e não retratando a realidade na variação de preço, já eventos imprevisíveis, como desastres naturais, afetam significativamente os preços. Recentemente, houve o caso da JBS [1], no Brasil, em que os seus gestores sabiam da delação premiada bombástica que seria publicada e se prepararam para lucrar com esse evento que o resto do mercado não

[1] De que JBS e BRF são acusadas? Comer carne é seguro? Entenda a operação da PF. 2017. URL: `https://brasil.elpais.com/brasil/2017/03/20/politica/1490036745_907943.html`

tinha conhecimento ainda. Felizmente, a CVM (Comissão de Valores Mobiliários) verificou a irregularidade e puniu a empresa. Porém, nem sempre é assim.

Há indícios de que as notícias influenciam na volatilidade e preço de ativos apenas em um curto espaço de tempo, por volta de 20 minutos antes e após o surgimento das mesmas(GIDÓFALVI, 2001), posteriormente estabilizando. Enquanto a teoria sobre a análise técnica indica que qualquer informação está presente no gráfico, o que inclui notícias, informações privilegiadas, especulação, entre outras. Portanto, os dados históricos são reprodutores fiéis do que ocorreu com um ativo, sendo mais confiáveis para se analisar do que apenas notícias.

Na *review* (KHADJEH NASSIRTOUSSI *et al.*, 2014) acerca de abordagens para previsão do mercado com base em textos, pode-se perceber que os casos em que foram utilizados dados numéricos juntamente com dados textuais obtiveram resultados melhores do que quando foram utilizados apenas dados textuais. É possível verificar que casos em que se utilizaram sistemas híbridos funcionaram melhor, enquanto os que se utilizavam apenas de notícias não conseguiram resultados tão bons. Já onde foram utilizados somente dados numéricos, os resultados foram os melhores possíveis (RACHLIN *et al.*, 2007). Ou seja, os dados numéricos por si só parecem ser mais seguros para realizarmos estudos, além de serem mais facilmente obtidos. Por isso, utilizar apenas dados históricos de ação ou criptomoedas parece ser uma opção melhor, restringindo o estudo apenas à análise de séries temporais, não só de dados financeiros, como também outros tipos.

1.2 Proposta

À luz dos estudos anteriores, este trabalho foca na análise exclusiva dos dados numéricos. Portanto, é importante saber qual metodologia e quais métodos computacionais utilizar. Na *survey* (ATSALAKIS e VALAVANIS, 2009) são descritas e comparadas dezenas de metodologias aplicadas a dados provenientes de diversos Mercados de Ações ao redor do mundo, com foco em Redes Neurais e Redes Neuro-Fuzzy.

Basicamente, a metodologia aplicada na maioria dos artigos é:

- Escolha dos dados e quais variáveis de entrada serão utilizadas.
- Definição do tamanho de treino e teste a partir da subdivisão dos dados originais. Atentando-se ao fato de serem séries temporais, portanto a ordem dos dados importa. Não podem existir dados de teste anteriores aos de treino e sua ordem inicial deve ser respeitada. Obs.: Muitas vezes utiliza-se uma parte dos dados de treino para serem usados como dados de validação e poder estimar como o modelo está se saindo.

- Pré-processamento: Normalização dos dados e outras técnicas.
- Aplicação de algum método computacional. Exemplos: Redes Neurais Artificiais (ANN), Máquina de Vetores de Suporte (SVM), Algoritmos Genéticos, entre outros. Alguns artigos também comparam seus resultados com estratégias convencionais (ATSALAKIS e K, 2013) como Modelos Autorregressivos (AR, ARMA e ARIMA), Regressão Linear e Multilinear (LR e MLR), Estratégia *Buy & Hold*, *Random Walk*, entre outros.
- Obtenção dos resultados e medição de desempenho, o qual pode ser classificado como estatístico ou não estatístico. Estatísticos seriam: Erro Quadrático Médio (MSE), Raiz do Erro Quadrático Médio (RMSE), Erro Absoluto Médio (MAE), dentre outros. Os não estatísticos seriam aqueles que medem a parte econômica das previsões, como o retorno de lucro anual ou o método *Hit Rate* que mede a porcentagem de acertos na previsão do modelo, por exemplo.

Na revisão é verificado que os métodos computacionais superam os métodos convencionais na maioria dos casos, retornando resultados melhores para *trading* e acurácia maior na previsão. A utilização de alguma Rede Neural Artificial para realizar as previsões parece uma escolha interessante.

Sabidamente, qualquer aplicação que lida com dados necessita de uma dedicação de tempo e esforço para garantir que os dados serão de qualidade (o que inclui limpeza de dados anômalos, os *outliers*, integração e transformação, e redução de dimensões) para, então, serem introduzidos em algum modelo computacional (PYLE, 1999; TAN *et al.*, 2005; MINING, 2006). O pré-processamento é uma fase muito importante e que, sendo bem feita, pode melhorar significativamente as previsões, inclusive em séries temporais complexas do mundo real, como os preços de ações ao longo do tempo, que normalmente são não-estacionárias e com volatilidade não uniforme (heteroscedasticidade) (SHANKER *et al.*, 1996; NAYAK *et al.*, 2012; OGASAWARA *et al.*, 2010; TSAY, 2005).

Neste trabalho, o foco é realizar melhorias e comparações na parte de transformação (especificamente normalização) dessa etapa do processo, para que se obtenha melhores previsões sem a necessidade de utilização de outros tipos de Redes Neurais, realizar um aumento do número de camadas ou de épocas na fase de treinamento. Em uma busca excessiva pela configuração ideal da Rede, resumindo-se a um procedimento de tentativa e erro (ATSALAKIS e VALAVANIS, 2009), o que pode ser muito custoso e nem sempre trazer bons resultados.

Os métodos de normalização comumente utilizados são o Min-Max, o Z-score e o Decimal. O primeiro normaliza os valores de entrada entre 0 e 1 ou -1 e 1. O segundo normaliza os valores de acordo com sua média e desvio padrão. No último, o ponto decimal dos valores é movido de acordo com o valor máximo absoluto da série.

Porém, esses métodos nem sempre podem ser aplicados em séries temporais reais, pois exigem certas premissas para que funcionem bem. Por exemplo, os métodos Decimal e Min-Max necessitam que se saiba o valor máximo e mínimo em uma série, o que nem sempre é possível. O que pode ser feito é assumir que os valores de máximo e mínimo estejam nos dados de treino, porém, nos dados de teste para séries temporais, ou seja, dados futuros, podem existir valores acima ou abaixo desses limites. Um exemplo seria uma ação variar durante um ano entre R$ 15,00 e R$ 17,00, porém futuramente atingir um valor de R$ 18,00. Para o método Min-Max existiria um "corte" na representação real da série ao se desnormalizar, já que a normalização não previa esse valor fora dos limites. Portanto, ao normalizar esse valor futuro, ele assumiria um valor de 1, representando o mesmo valor que os R$ 17,00 do treino, e mal representando a série real. Já o Z-score pode ser aplicado para séries estacionárias (GUJARATI, 2008; KENDALL, 1976), ou seja, séries temporais que possuam média, variância e auto-correlação constantes ao longo do tempo. Porém, estacionariedade não ocorre na maioria das séries temporais do mundo real do Mercado Financeiro (TSAY, 2005), já que suas propriedades estatísticas variam ao longo do tempo.

Apesar de existirem formas de transformar séries não-estacionárias em estacionárias, através de manipulações matemáticas, como a diferenciação (CRYER e CHAN, 2008), isso nem sempre é apropriado (NELSON e PLOSSER, 1982; PIERCE, 1977), pois se removem informações de longo prazo dos dados, como tendência (OGASAWARA *et al.*, 2010).

Uma abordagem tradicionalmente usada para resolver os problemas das técnicas de normalização citadas anteriormente é o uso do *Sliding Windows*(SW) ou "janelas deslizantes"(HAYKIN *et al.*, 2009). Essa técnica divide a série temporal em janelas de tamanho w, extraindo propriedades estatísticas de cada janela w e realizando normalização para cada uma delas. Essa técnica funciona bem para séries temporais que possuem volatilidade uniforme ao longo do tempo (homocedásticas), porém a maioria não a possui (TSAY, 2005), sendo chamadas de heteroscedásticas (GUJARATI, 2008). Seu ponto fraco é a sua baixa representatividade da volatilidade global da série temporal, ao assumir a mesma volatilidade para cada janela w.

Eduardo Ogasawara apresentou uma técnica, a Normalização Adaptativa (AN), que é uma variação da *Sliding Windows*, visando resolver o problema da representação de volatilidade global (OGASAWARA *et al.*, 2010). Sua principal diferença é que a AN transforma a série em uma sequência de dados de onde estatísticas globais podem ser retiradas dos dados de treino e consideradas no processo de normalização. Os resultados obtidos no seu artigo demonstraram que essa técnica melhora a acurácia de uma Rede Neural tanto em curto como em longo prazo.

O Objetivo desse trabalho é apresentar propostas que tragam uma melhoria a

essa técnica, de forma que se evite uma falha identificada para séries temporais específicas, as quais possuem médias móveis de valor 0, como é o caso da série $f(x) = \cos x$, e também se possa aumentar a acurácia, portanto evoluindo o estado da arte. Visa-se também aplicar essa técnica em previsões de curtíssimo prazo, por minuto, enquanto no artigo a técnica é testada em modelos de previsões diárias ou trimestrais, assim como aplicar em diferentes domínios (como indício de chuvas em uma cidade ou volume de vazões em uma hidrelétrica), e comparar com as outras 4 técnicas de normalização citadas, além da técnica convencional de modelo auto-regressivo (AR), integrados (I) e de médias móveis(MA), ARIMA, comumente utilizada como *baseline* em trabalhos desse tipo (TSAY, 2005; SALLES *et al.*, 2017).

1.3 Contribuições

Esse trabalho possui as seguintes contribuições:

1. Um estudo da aplicação da Normalização Adaptativa para dados por minuto, ou seja, para aplicação em *Day Trade*.
2. Duas novas normalizações baseadas na Normalização Adaptativa, de modo a corrigir sua falha relativa a médias móveis de valor nulo.
3. Estudo comparativo de desempenhos dessas duas novas normalizações com a Normalização Adaptativa e outras normalizações presentes na literatura.
4. Análise do desempenho das novas normalizações propostas em diferentes tipos de *datasets*.

1.4 Organização

Essa dissertação está organizada em 6 capítulos, dos quais o presente é o primeiro e introduz o trabalho. No capítulo 2, descrevem-se os principais conceitos teóricos envolvidos, incluindo-se o funcionamento das principais normalizações presentes na literatura e da Normalização Adaptativa. No capítulo 3 são propostas duas novas Normalizações baseadas na Normalização Adaptativa, com o objetivo de corrigir o problema das médias móveis nulas em séries temporais. No capítulo 4, descrevem-se os objetivos dos experimentos, assim como a metodologia utilizada, descrição dos *Datasets* usados, configuração dos experimentos, métricas e tecnologia utilizada, também exibem-se os resultados obtidos para cada *Dataset*, e realiza-se uma análise dos mesmos. Enfim, no capítulo 5, exibem-se as considerações finais, contribuições e propõe-se trabalhos futuros.

Capítulo 2

Fundamentação Teórica

2.1 Séries Temporais

Séries Temporais são observações feitas sequencialmente ao longo do tempo, ou seja, a ordem dos dados observados é essencial, pois observações próximos são dependentes de forma ordenada (WOOLDRIDGE, 2009). Muitas vezes uma série temporal pode ser definida por uma função matemática em relação ao tempo, como é o caso da função senoidal, representada na figura 2.1. Nesse caso considera-se que ela possui apenas componente determinístico.

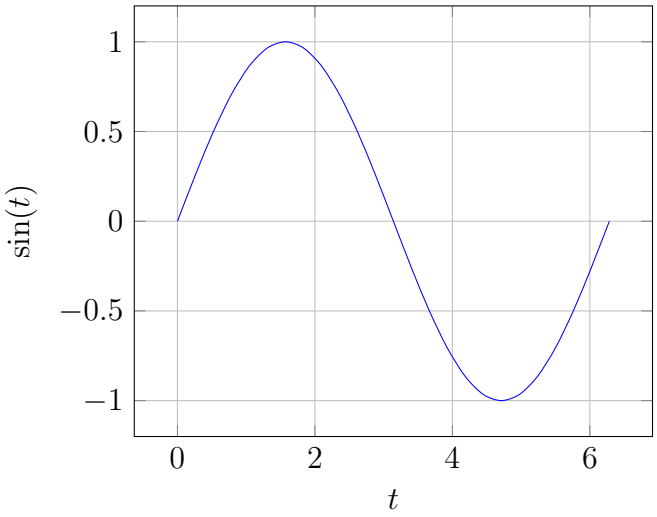

Figura 2.1: Função Senoidal ao longo do tempo

Séries temporais do mundo real, como os preços de criptomoedas ao longo do tempo, normalmente possuem componentes estocásticos, ou seja, componentes aleatórios gerados por um processo estocástico. Um exemplo de representação de uma série com componente aleatório ϵ_t está na equação 2.1, também possuindo uma parte determinística. O componente estocástico introduz incerteza à série.

$$y_t = \underbrace{a + b \cdot x_t}_{\text{determinístico}} + \underbrace{\epsilon_t}_{\text{estocástico}} \tag{2.1}$$

Normalmente as séries temporais são analisadas a partir de seus principais movimentos como tendência, sazonalidade, variações aleatórias, dentre outras. Uma forma que se pode classificá-las é em Séries estacionárias e não-estacionárias.

2.1.1 Séries estacionárias

São séries temporais que flutuam em torno de uma mesma média ao longo do tempo e que suas propriedades estatísticas, como variância e auto correlação se mantenham constantes. Caso ela possua componente estocástico, ou aleatório, ele deve ser retirado de uma distribuição normal com média zero e variância finita, dessa forma a estacionariedade continuará sendo respeitada. Um exemplo de série estacionária é a senoidal referenciada anteriormente 2.1.

2.1.2 Séries não-estacionárias

Basicamente são as séries que não possuem padrão definido, ou seja, suas propriedades estatísticas variam ao longo do tempo. Na Figura 2.2 está um exemplo de série não-estacionária, que representa as cotações de fechamento por minuto dos mini contratos de dólar Futuro para o primeiro dia do mês de Setembro de 2016. Essa série também apresenta heteroscedasticidade, ou seja, volatilidade não uniforme ao longo do tempo (TSAY, 2005), como se pode notar no gráfico que o período da manhã, entre 9:00 e 12:00, há uma volatilidade grande, enquanto no meio da tarde a volatilidade é bem menor, entre 14:00 e 16:00.

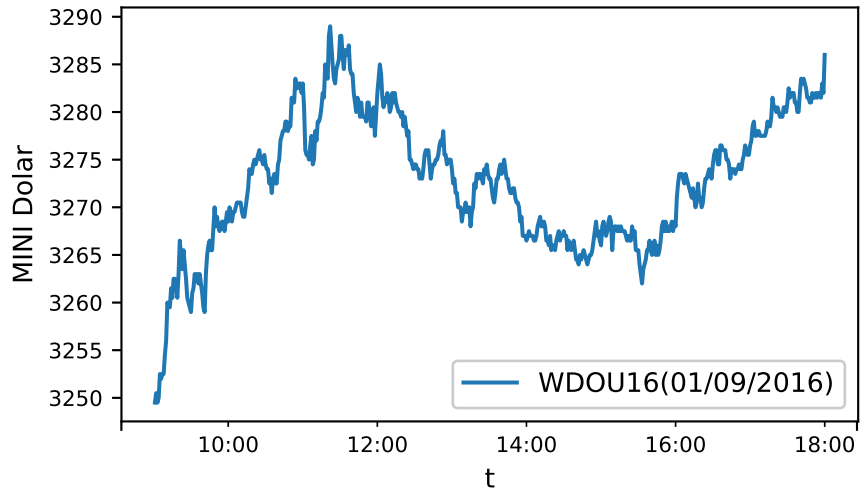

Figura 2.2: Cotação de Fechamento por minuto de Mini Contratos de Dólar Comercial Futuro (WDOU16) para dia 1º de Setembro de 2016

Como a maioria das séries temporais presentes no Mercado Financeiro, ela possui não-estacionariedade (TSAY, 2005). Obs.: a cotação do Mini contrato de dólar Futuro é expressa em reais por US$ 1000,00, portanto 3260 representa uma cotação de R$ 3,26 para cada dólar comercial [1].

Cabe ressaltar que apesar de existir esse conceito de estacionariedade e não-estacionariedade, podem-se definir graus de estacionariedade. Por exemplo, uma série pode ser classificada como fracamente estacionária se nem a média nem as auto-covariâncias dependem do tempo, porém a variância é finita para que ainda seja estacionária. Existe também a estacionariedade estrita, onde a distribuição conjunta não depende do tempo (TSAY, 2005).

2.1.3 Volatilidade

Volatilidade, em termos financeiros, é o grau de variação de preço de um ativo sobre o tempo, como medido pelo desvio padrão dos retornos logarítmicos. Retorno logarítmico é definido na equação 2.2, sendo V_f valor final e V_i valor inicial.

$$R = \ln\left(\frac{V_f}{V_i}\right) \tag{2.2}$$

[1] Futuro Míni de Taxa de Câmbio de Reais por Dólar Comercial. 2018. URL: `http://www.bmfbovespa.com.br/pt_br/produtos/listados-a-vista-e-derivativos/moedas/futuro-mini-de-taxa-de-cambio-de-reais-por-dolar-comercial.htm`

2.2 Redes Neurais Multicamadas (MLP)

As Redes Neurais Multicamadas ou *Multilayer Perceptron* (MLP) são parte de uma classe de Redes Neurais *feedforward* e estão entre as mais simples das Redes Neurais (HASTIE *et al.*, 2009), sendo conhecida como um Aproximador Universal, ou seja, consegue representar qualquer função mensurável a qualquer grau de acurácia desejada (HORNIK *et al.*, 1989). Na prática, a aplicação ao mundo real não é simples, exigindo uma busca incessante pelos melhores parâmetros (LECUN *et al.*, 1998; BLUM e RIVEST, 1988; GLOROT e BENGIO, 2010), dentre eles o número de camadas, de neurônios por camada, função de ativação, assim como a aplicação de métodos de normalização (SHANKER *et al.*, 1996). Essa é a Rede utilizada nesse trabalho, sendo representada a seguir na Figura 2.3 com apenas uma camada intermediária (*Hidden layer*) e 4 entradas na *Input layer* e uma saída no *Output layer*.

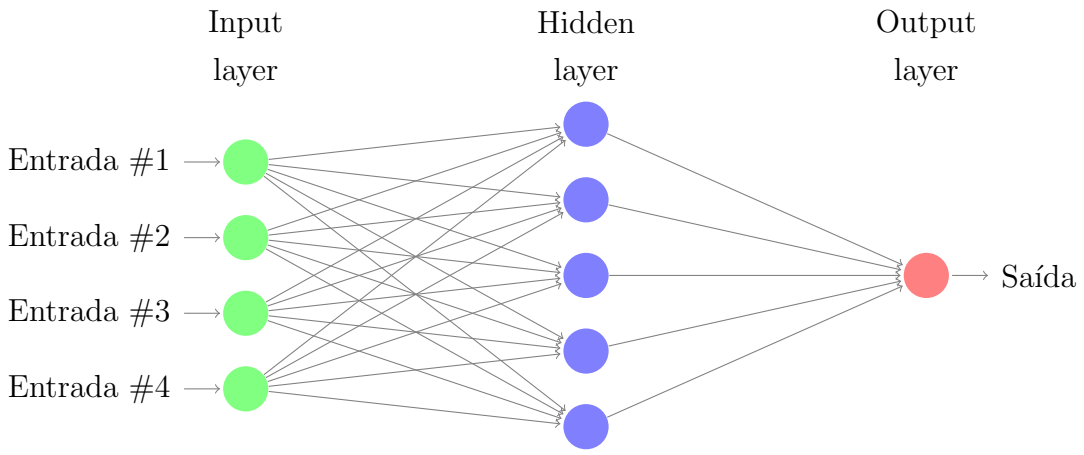

Figura 2.3: Exemplo de Rede Neural MLP

Nesse trabalho, as 4 entradas representadas na Rede Neural seriam uma sequência dos valores de fechamento de 4 minutos conhecidos e a saída seria o valor previsto de fechamento no minuto seguinte. Ressaltando que as entradas estariam devidamente normalizadas antes de serem inseridas na Rede, portanto a saída estaria dentro da mesma escala de normalização.

2.3 Métodos de Normalização

Nessa seção descrevem-se os métodos de normalização utilizados para comparação neste trabalho.

2.3.1 Min-Max

O método Min-Max normaliza os valores de um atributo A de acordo com seus valores mínimo e máximo, convertendo um valor original a do atributo A em a', como pode ser visto na equação (2.3).

$$a' = (high - low)\frac{a - \min A}{\max A - \min A} + low \tag{2.3}$$

Seu principal problema é que não se sabe os valores mínimo e máximo dos dados de teste numa série temporal. Então, uma forma simples de se contornar isso é considerar os valores mínimo ($\min A$) e máximo ($\max A$) dos dados de treino para a normalização, e mapear os valores nos dados de teste que estejam acima de $\max A$ como *high* e abaixo de $\min A$ como *low*. Essa estratégia acaba por permitir uma perda relevante de informação e uma concentração de valores em determinadas partes do intervalo normalizado (TAN *et al.*, 2005), causando esforço computacional maior e perda de qualidade nas previsões (OGASAWARA *et al.*, 2010; SHALABI e SHAABAN, 2006; SOLA e SEVILLA, 1997).

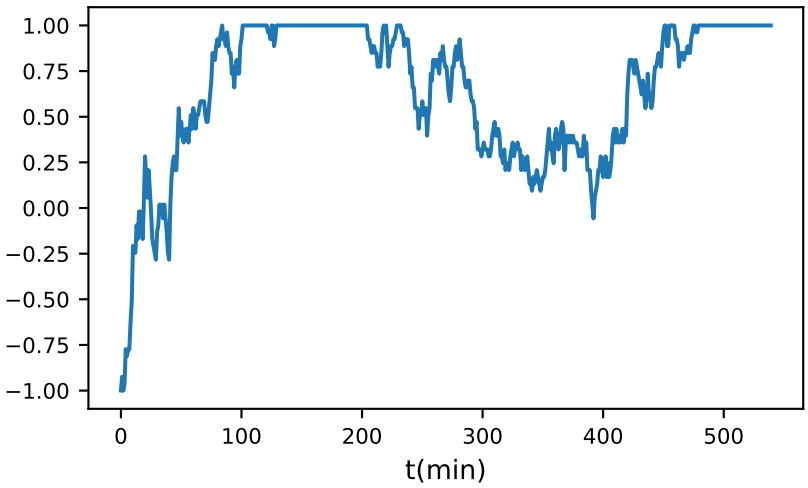

Figura 2.4: Normalização Min-Max a partir da amostra entre 9:00 e 10:30 do dia 1º de Setembro de 2016

Na Figura 2.4 se apresenta o problema da concentração de valores em *high* ou 1 devido ao "corte"causado pela limitação da amostra. A amostra (que seria usada nos dados de treino para uma aprendizagem de Rede Neural, por exemplo) representando as cotações de Fechamento por minuto de WDOU16 entre 9:00 e 10:30 da série temporal, visualizada na Figura 2.2, apresenta valor mínimo ou *low* para toda a série, portanto não havendo "cortes"por baixo e sendo capaz de normalizar sem perda de informação. Não se pode afirmar o mesmo para o valor máximo ou *high* da amostra, já que existem valores superiores a ele após o período amostral,

causando o "corte"após se realizar a normalização Min-Max e a devida perda de informação da série original. Em resumo, no caso dessa série temporal existem perdas e concentração de valores em determinados pontos superiores da série, porém não existem perdas para pontos inferiores, mas isso varia de acordo com o tamanho da amostragem, tendência da série, entre outros fatores.

2.3.2 *Z-score*

Na normalização *Z-score*, os valores de um atributo A são normalizados de acordo com suas média e desvio padrão, ou seja, não exige conhecimento de valores máximo e mínimo da série, como no caso da normalização Min-Max. Um valor a de A é normalizado para a', como pode ser visto na equação (2.4) (onde $\mu(A)$ é a média do atributo A e $\sigma(A)$ é seu desvio padrão).

$$a' = \frac{a - \mu(A)}{\sigma(A)} \tag{2.4}$$

Esse método é interessante para séries estacionárias, já que o desvio padrão e a média se mantêm uniformes ao longo do tempo, mesmo que os valores de máximo e mínimo não sejam conhecidos. Porém, não lida bem com séries não-estacionárias pois possuem suas propriedades estatísticas variantes ao longo do tempo, as quais são estudadas nesse trabalho. Como pode ser observado na Figura 2.5, ao se considerar média e desvio padrão de diferentes partes da série amostral, a normalização traz resultados diferentes. Esse é um problema que não ocorreria caso a série fosse estacionária ao longo de todo o tempo observado.

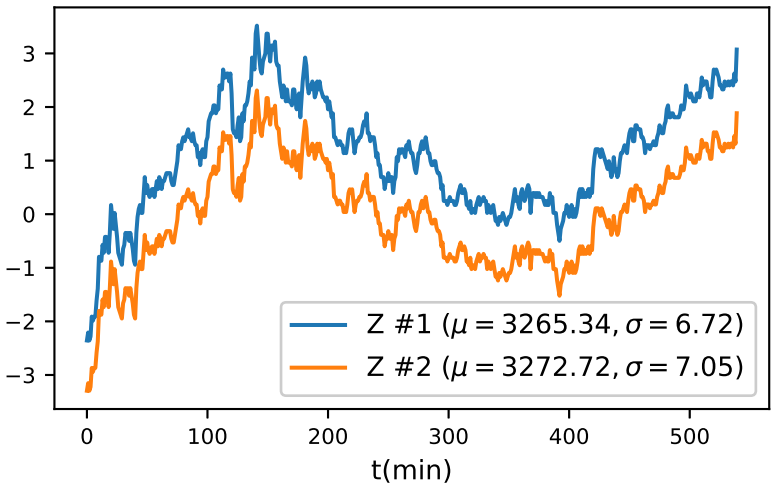

Figura 2.5: Normalizações por *Z-score*: Z #1 utilizando média e desvio padrão para o período amostrado entre 9:00 e 10:30 do dia 1º de Setembro de 2016 e Z #2 utilizando média e desvio padrão para todo o período do dia 1º de Setembro de 2016

2.3.3 Decimal

A normalização por escala decimal realiza o reposicionamento do ponto decimal dos valores de um atributo A de acordo com o seu valor máximo absoluto. Ou seja, um valor a de A é normalizado para a' ao ser dividido por 10^d, como pode ser visto na equação (2.5), onde d é o menor número inteiro para que $\max(|a'|) < 1$.

$$a' = \frac{a}{10^d} \tag{2.5}$$

Por exemplo, dados os valores 400 e 1200, o cálculo do d seria baseado no valor máximo, ou seja, no 1200 (a), que, ao se mover o ponto decimal ficaria 0.12 (a') e $d = 4$, enquanto o valor 400 (a) ficaria como 0.04 (a').

Esse método possui o mesmo problema que o Min-Max, pois se necessita saber o valor máximo baseado em uma amostra, sendo que existe a possibilidade de existirem valores superiores a esse em dados de Fechamento posteriores, como está exemplificado na Figura 2.6.

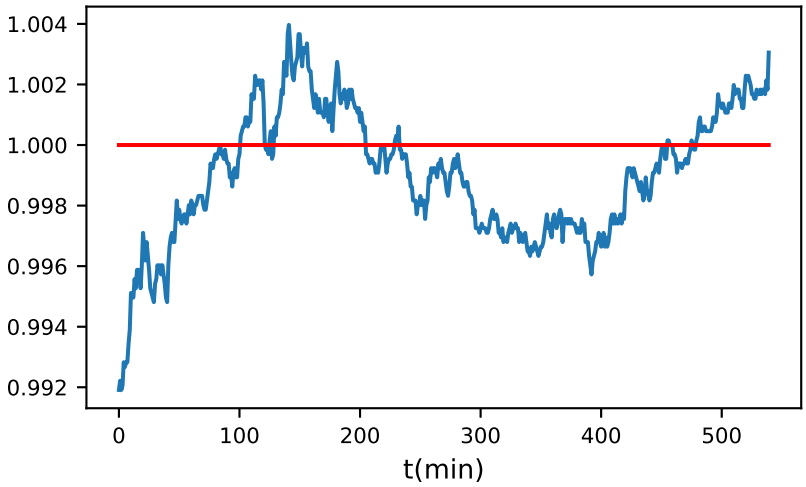

Figura 2.6: Normalização Decimal a partir da amostra entre 9:00 e 10:30 do dia 1º de Setembro de 2016

É possível observar que nesse tipo de normalização há um espectro muito pequeno entre o valor máximo e mínimo pós normalizar, sendo o mínimo de 0.992 e máximo de 1 (considerando a amostragem e o corte) no caso da Figura 2.6. Isso ocorre por que os valores originais são da ordem de milhares, portanto o valor $\max(|a'|)$ seria de 3276.0 para a amostra nesse exemplo, cujo representa o valor de 1 após ser normalizado.

2.3.4 *Sliding Window*

O fundamento por trás dessa técnica está em não considerar a série temporal inteira para a normalização. Ao invés disso, se divide os dados em janelas deslizantes de tamanho w, ou seja, janelas separadas por períodos unitários, com as janelas se sobrepondo uma a uma, como em um ato de deslizar sobre a série temporal. Para cada janela w, apenas suas propriedades estatísticas são utilizadas para a normalização (TAN *et al.*, 2005; LI e LEE, 2009). A ideia por trás dessa abordagem se justifica por normalmente as decisões serem tomadas baseadas em dados recentes, no caso do Mercado de Ações, por exemplo.

Essa técnica possui a vantagem de sempre normalizar por Min-Max os dados dentro do intervalo desejado, entre 0 e 1 ou -1 e 1, porém tem o lado negativo de assumir que a série temporal possui volatilidade uniforme ao longo do tempo, o que não é muito comum em séries temporais do mundo real (TSAY, 2005; GUJARATI, 2008; HULL, 2010). Na Figura 2.7, pode-se observar esse fenômeno: duas janelas, de diferentes volatilidades originalmente, são normalizadas e ficam com a mesma volatilidade, gerando as janelas deslizantes SW #1 normalizada a partir de um período com baixa volatilidade e SW #2 a partir de um com alta volatilidade, ambas com uma janela w de 20 períodos de 1 minuto. A janela SW #1 é normalizada a partir do período entre 9:50 e 10:10 e SW #2 a partir do período entre 11:00 e 11:20 do dia 1º de Setembro de 2016, para observar a volatilidade de ambos períodos, basta checar na Figura 2.2.

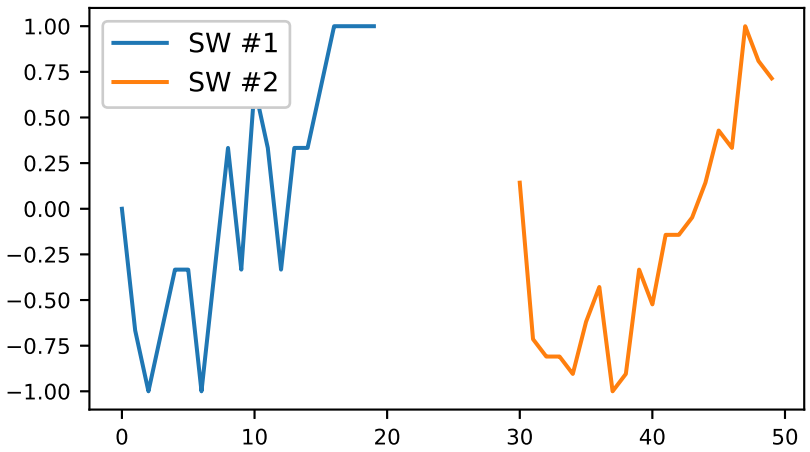

Figura 2.7: Janelas de tamanho 20 períodos normalizadas por *Sliding Window*: SW #1 para o período de baixa volatilidade entre 9:50 e 10:10 e SW #2 para o período de alta volatilidade entre 11:00 e 11:20 do dia 1º de Setembro de 2016

2.4 Normalização Adaptativa

Os métodos discutidos anteriormente são relativamente eficazes em séries temporais estacionárias. O método *Sliding Window* tenta superar as limitações dos outros métodos, ao ser capaz de lidar com séries não-estacionárias, porém apenas com volatilidade uniforme. Como as séries temporais estudadas nesse trabalho são não-estacionárias com heteroscedasticidade, elas não são capazes de representá-las corretamente após a respectiva normalização ser utilizada.

A Normalização Adaptativa (AN) apresenta uma abordagem para ser aplicada a esse tipo de série temporal (OGASAWARA *et al.*, 2010). Seu processo de normalização pode ser subdividido em três estágios:

1. Transformar a série não-estacionária em uma sequência estacionária, criando uma sequência de janelas deslizantes desconectadas, ou seja, não sobrepostas.

2. Remoção de *Outliers*.

3. Normalização dos dados.

Os dados resultantes desse processo de normalização podem ser usados para aplicação em um método de Aprendizagem de Máquinas, como Redes Neurais. E, obviamente, após as previsões serem feitas deve ocorrer uma desnormalização e destransformação para que os valores previstos sejam mapeados para valores compatíveis com o da série temporal original.

2.4.1 Transformação dos dados

A Normalização Adaptativa se utiliza de médias móveis no processo de transformação, a série, originalmente não-estacionária é transformada em uma sequência estacionária. Primeiramente, as médias móveis (CRYER e CHAN, 2008) são calculadas para a série temporal original, depois elas são utilizadas para criar uma nova sequência estacionária, sendo divididas em janelas deslizantes desconectadas. Na Tabela 2.1 estão os dados utilizados para exemplificar o processo de normalização.

Tabela 2.1: Exemplo representando os 15 minutos iniciais de abertura de mercado do dia 1º de Setembro de 2016 para o papel WDOU16 e suas respectivas médias móveis exponenciais de período 5

t(m): i	WDOU16: S	EMA5: $S_e^{(5)}$
0	3249.5	–
1	3250.5	–
2	3249.5	–
3	3250.0	–
4	3252.5	3250.89
5	3252.0	3251.30
6	3252.5	3251.72
7	3252.5	3251.99
8	3254.5	3252.85
9	3256.0	3253.92
10	3260.0	3255.97
11	3259.5	3257.94
12	3259.5	3259.13
13	3261.5	3259.59
14	3260.5	3260.56
15	3262.5	3261.21

Médias móveis vem sendo usadas exaustivamente na área de finanças e econometria (TSAY, 2005; CHATFIELD, 2004). Analistas técnicos buscam utilizá-las para tentar identificar mudanças de tendências e padrões nas séries temporais que representam fechamento do preço de ações, por exemplo. Também é possível se identificar mudanças no comportamento da série com mais clareza, devido a redução do efeito de ruído (MOON e KIM, 2007), como pode ser observado na Figura 2.8, em que, quanto maior o número de períodos utilizados para se calcular a média móvel, menor será perceptível pequenas variações da série temporal original, ou seja, menor será a influência de ruídos, porém, pior será representada com fidelidade a série temporal original. Na 2.8, quando as Médias Móveis são calculadas por 5 períodos, a sequência EMA5 parece acompanhar mais fielmente a série temporal original, enquanto a EMA21 não sofre grandes efeitos dos ruídos, seguindo a tendência da série original de forma mais suave.

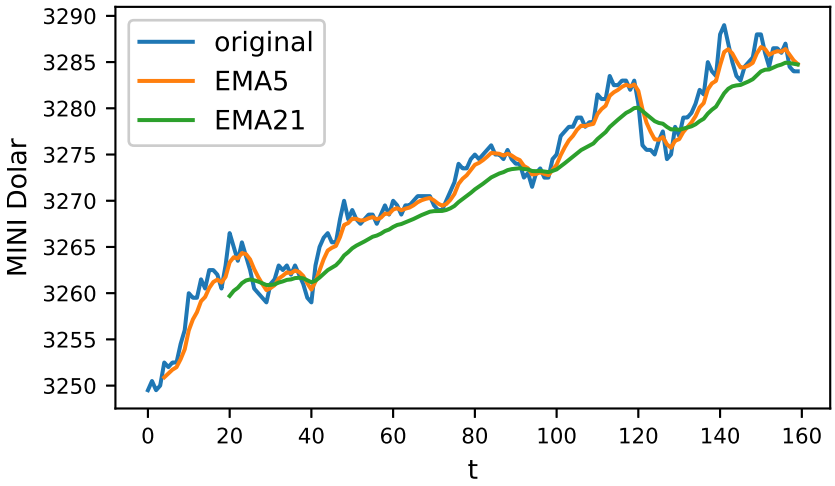

Figura 2.8: Médias Móveis Exponenciais de 5 e 21 períodos

Elas também possuem a característica de lidar implicitamente com a inércia (GUJARATI, 2008). Inércia é um conceito físico e matemático que representa a resistência que um objeto oferece para mudar seu estado de movimento. Um objeto que está em repouso tende a permanecer em repouso, ou se estiver em movimento retilíneo uniforme tende a permanecer nesse estado, de acordo com a Primeira Lei de Newton (NEWTON, 1999). Em séries temporais, a inércia é uma propriedade importante que permite lidar com o dilema da estabilidade-plasticidade (GROSSBERG, 1988). Esse dilema é uma limitação bem conhecida para redes neurais artificiais e biológicas. Basicamente, o aprendizado em sistemas paralelamente distribuídos requerem plasticidade para a integração de novos conhecimentos, mas também estabilidade para prevenir o esquecimento de conhecimentos anteriores (MERMILLOD *et al.*, 2013).

Na AN, em uma média móvel (*Moving Average* ou MA) de ordem k, k corresponde ao número de períodos usados para introduzir inércia à nova sequência estacionária. Mas convertem uma determinada sequência S em uma nova sequência $S^{(k)}$, a qual representa a média de k valores consecutivos da sequência S, como pode ser observado na Tabela 2.1. Ou seja, dada uma sequência $S = \{S[0], S[1], \ldots, S[n]\}$ de tamanho n e MA de ordem $k(1 \leq k \leq n)$, i-ésimo valor $S^{(k)}[i](k-1 \leq i \leq n)$ é definido como uma média dos valores da subsequência $S[i-k+1:i]$. No caso exposto na Tabela 2.1, os 4 primeiros elementos da MA de ordem $k = 5$, não são passíveis de serem computados, já que não se sabe os valores por completo da subsequência composta pelos k períodos anteriores da série original S, somente quando $i = (k-1)|_{k=5} = 4$, pois assim é possível realizar o cálculo da média com todos os valores conhecidos da subsequência representada por $S = \{S[0], S[1], S[2], S[3], S[4]\}$.

Segundo o artigo (OGASAWARA *et al.*, 2010), dois tipos de médias móveis po-

dem ser usadas no processo de Normalização Adaptativa: Média Móvel Simples (*Simple Moving Average* ou SMA) ou Média Móvel Exponencial (*Exponential Moving Average* ou EMA) (JIANG e SZETO, 2003). A SMA representa uma sequência de médias sem pesos, como pode ser visto na equação (2.6).

$$S_s^{(k)}[i] = \frac{1}{k} \sum_{j=i-k+1}^{i} S[j], \forall i / k - 1 \leq i \leq n \tag{2.6}$$

Enquanto a EMA representa um sequência de médias com pesos decrescentes exponencialmente, ou seja, pesos maiores para observações mais recentes, sendo os pesos reduzidos por um fator de suavização $\alpha (0 \leq \alpha \leq 1)$, normalmente calculado com base na ordem k da EMA, da seguinte forma: $\alpha = \frac{2}{(k+1)}$. Ela pode ser definida de forma recursiva, segundo a equação (2.7).

$$S_e^{(k)}[i] = \begin{cases} S_e^{(k)}[i] = S[i], & i = 1 \\ \alpha \cdot S[i] + (1 - \alpha) \cdot S_e^{(k)}[i-1], & i > 1 \end{cases} \tag{2.7}$$

Se a ordem k for 5, como visto na Tabela 2.1, o α vai ter um valor aproximado de 0.333. Como exemplo da aplicação da equação (2.7), considerando $i = 5$, substituindo os valores na equação, se pode calcular: $S_e^{(k)}[5] = 0.333 \cdot S[5] + 0.667 \cdot S_e^{(k)}[4] = 0.333 \cdot 3252.0 + 0.667 \cdot 3250.89 = 3251.26 \equiv 3251.3$. Outra forma de representar a equação (2.7) é a variante (2.8) supondo uma série infinita. Tal variação foi utilizada no cálculo das EMAS da Tabela 2.1 e deste trabalho, podendo acarretar valores um pouco diferentes, já que a série é finita.

$$S_e^{(k)}[i] = \frac{S[i] + (1-\alpha)S[i-1] + (1-\alpha)^2 S[i-2] + ... + (1-\alpha)^i S[0]}{1 + (1-\alpha) + (1-\alpha)^2 + ... + (1-\alpha)^i} \tag{2.8}$$

A comprovação de que ambas são equivalentes está na equação (2.9), caso a série original tenha um histórico infinito. O denominador se torna uma série geométrica com termo inicial 1 e razão $1 - \alpha$.

$$\begin{aligned} y_t &= \frac{x_t + (1-\alpha)x_{t-1} + (1-\alpha)^2 x_{t-2} + ...}{\frac{1}{1-(1-\alpha)}} \\ &= [x_t + (1-\alpha)x_{t-1} + (1-\alpha)^2 x_{t-2} + ...]\alpha \\ &= \alpha x_t + [(1-\alpha)x_{t-1} + (1-\alpha)^2 x_{t-2} + ...]\alpha \\ &= \alpha x_t + (1-\alpha)[x_{t-1} + (1-\alpha)x_{t-2} + ...]\alpha \\ &= \alpha x_t + (1-\alpha) y_{t-1} \end{aligned} \tag{2.9}$$

Após realizar a criação da sequência de médias móveis, se pode transformar

a série original não-estacionária em uma sequência estacionária dividida em janelas deslizantes desconectadas pela Normalização Adaptativa (OGASAWARA *et al.*, 2010). Dada uma sequência S de tamanho n, sua referida média móvel de ordem k, $S^{(k)}$ de tamanho $n - k + 1$, e um tamanho de janela w, novas subsequências r_i podem ser definidas como na equação (2.10).

$$r[i] = \frac{S[i, \ldots, (i + w - 1)]}{S^{(k)}[i]}, \forall i / w \leq i \leq (n - w + 1) \tag{2.10}$$

São geradas, portanto, $n - w + 1$ janelas deslizantes desconectadas r_i. Para cada janela, todo denominador será o mesmo ($S^{(k)}[i]$). Esse fator é importante para preservar a tendência original da série temporal e trazer a mesma inércia para todos os valores na mesma janela. Cada janela possui w valores de entrada e um valor de saída, na prática, se realiza o cálculo com o mesmo denominador para uma janela de tamanho $w + 1$ para normalizar também o valor a ser previsto, quando se objetiva treinar uma Rede Neural, por exemplo.

Na Tabela 2.2, se encontram duas janelas r_i apenas contendo entradas, com tamanho $w = 20$: AN # 1 representando o período entre 9:50 e 10:10 e AN #2 para o período entre 11:00 e 11:20 do dia 1º de Setembro de 2016. Janelas também chamadas de r_{50} e r_{120}, pois seu início representa o minuto 50 e 120, respectivamente, da série original. Nesse caso, as janelas são representadas pelas subsequências: $r_{50} = \frac{S[50,\ldots,69]}{S_e^{(5)}[50]}$ e $r_{120} = \frac{S[120,\ldots,139]}{S_e^{(5)}[120]}$. Nota-se o mesmo denominador para todos os 20 elementos da janela, representando a MA referente ao primeiro elemento de cada subsequência ou janela r_i. Caso se estivesse considerando também a saída para o treinamento de uma Rede Neural, o denominador também seria o mesmo, mas se consideraria um tamanho de $w+1$, representando a janela de entradas mais a saída, o que seria tratado como previsão.

Tabela 2.2: Exemplo de duas janelas deslizantes desconectadas r_i

t(m): i	AN #1: r_{50+i}	AN #2: r_{120+i}
0	1.00028826	0.99957784
1	0.99998227	0.99820668
2	0.99982928	0.99805433
3	0.99998227	0.99805433
4	1.00013527	0.99790197
5	1.00013527	0.99835903
6	0.99982928	0.99866373
7	1.00013527	0.99774962
8	1.00044126	0.99790197
9	1.00013527	0.99881608
10	1.00059426	0.99851138
11	1.00044126	0.99912079
12	1.00013527	0.99912079
13	1.00044126	0.99927314
14	1.00044126	0.99957784
15	1.00059426	1.00003489
16	1.00074725	0.99988254
17	1.00074725	1.000949
18	1.00074725	1.0006443
19	1.00074725	1.00049195

O tipo de MA utilizada na AN e sua ordem irá variar de acordo com as características de cada série temporal. No artigo, é sugerida uma técnica de ajuste de nível para todas as combinações de Tipo de MA e ordem k, a que possuir o melhor ajuste para todas as janelas desconectadas é selecionada (OGASAWARA *et al.*, 2010). Porém, essa técnica se mostrou ineficiente e não é utilizada nesse trabalho.

2.4.2 Remoção de *Outliers*

A segunda etapa consiste em remover *outliers* (PYLE, 1999; TAN *et al.*, 2005; CHOY, 2001). É uma etapa importante na fase de pré-processamento e para análise de séries temporais. O principal problema que ocorre no processo de normalização é quando os *outliers* aparecem nos limites extremos de uma série temporal, levando a valores incoerentes de máximos e/ou mínimos, afetando assim a estatística global da série, afetando também a qualidade da normalização, já que alguns dados podem se concentrar em determinados espaços do intervalo normalizado.

Para evitar esse tipo de problema, um método baseado em *Box plots* (TUKEY,

1977) pode ser utilizado para remover esses elementos indesejáveis dos dados. O método remove qualquer valor menor que o primeiro quartil ($Q1$) menos uma constante α multiplicado pelo valor de intervalo interquartil (IQR), e qualquer valor maior que o terceiro e último quartil ($Q3$) mais α vezes o valor de intervalo interquartil (IQR). Normalmente o valor adotado para α é 1.5, mas também é usado o valor de 3.0, caso seja desejada uma remoção de *outliers* mais branda, removendo apenas *outliers* extremos (KVANLI *et al.*, 2006). Ou seja, todos os valores que não estão no intervalo $[Q1 - \alpha \times IQR, Q3 + \alpha \times IQR]$ são considerados *outliers*.

Na AN, se qualquer valor considerado *outlier* estiver presente em alguma janela, a tal é eliminada do treinamento e do conjunto de testes. A eliminação do conjunto de testes faz-se necessária já que seria mais difícil de se prever, então, ao se formar janelas durante os testes, pode-se decidir por eliminar as que não se encaixam.

2.4.3 Normalização dos dados

A Normalização Adaptativa utiliza o método min-max para normalizar os valores das janelas r_i no intervalo entre -1 e 1, porém de uma forma diferente do usado no método tradicional *Sliding Windows*. Todas as janelas geradas r_i são exploradas para que se obtenha estatísticas globais, incluindo valores de máximo e mínimo globais, e usar esses valores como entrada pro método de normalização min-max. Importante ressaltar que os valores obtidos na remoção de *outliers* por *Box plots* são considerados máximo e mínimo caso existam valores maiores e menores que esses presentes em alguma janela, respectivamente. Já no método *Sliding Windows*, como discutido anteriormente, são consideradas propriedades estatísticas para cada janela independentemente.

A tabela 2.3 representa exemplo de duas janelas r_i após realização da normalização min-max, considerando os valores máximo e mínimo apenas do dia 1º de Setembro de 2016. Na Figura 2.9 é possível visualizar essas janelas normalizadas por AN com EMA de ordem $k = 5$ e $w = 20$ e compará-las com as janelas normalizadas com o método *Sliding Windows* com $w = 20$, representado na Figura 2.7. Ambas estão representando os mesmos períodos e as mesmas janelas a partir da série temporal original, porém, percebe-se que os problemas observados no método SW não se repetem no método AN. Originalmente, o período da série temporal original que posteriormente gera SW #1 possui uma volatilidade baixa ao se comparar com SW #2, a qual é mantida na AN #1 em relação a AN #2. Na SW há também um "esticamento"dos valores para cada janela para que fiquem no intervalo entre -1 e 1, enquanto na AN isso não ocorre, a volatilidade global obtida da série temporal original é respeitada.

Tabela 2.3: Exemplo de duas janelas r_i normalizadas com a Normalização Adaptativa com EMA de ordem $k = 5$ e $w = 20$ no intervalo entre -1 e 1

t(m): i	AN #1: r_{50+i}	AN #2: r_{120+i}
0	-0.2313242	-0.42517939
1	-0.31482095	-0.79933169
2	-0.35656932	-0.84090417
3	-0.31482095	-0.84090417
4	-0.27307257	-0.88247664
5	-0.27307257	-0.75775921
6	-0.35656932	-0.67461425
7	-0.27307257	-0.92404912
8	-0.18957582	-0.88247664
9	-0.27307257	-0.63304178
10	-0.14782744	-0.71618673
11	-0.18957582	-0.54989682
12	-0.27307257	-0.54989682
13	-0.18957582	-0.50832434
14	-0.18957582	-0.42517939
15	-0.14782744	-0.30046195
16	-0.10607907	-0.34203443
17	-0.10607907	-0.05102709
18	-0.10607907	-0.13417204
19	-0.10607907	-0.17574452

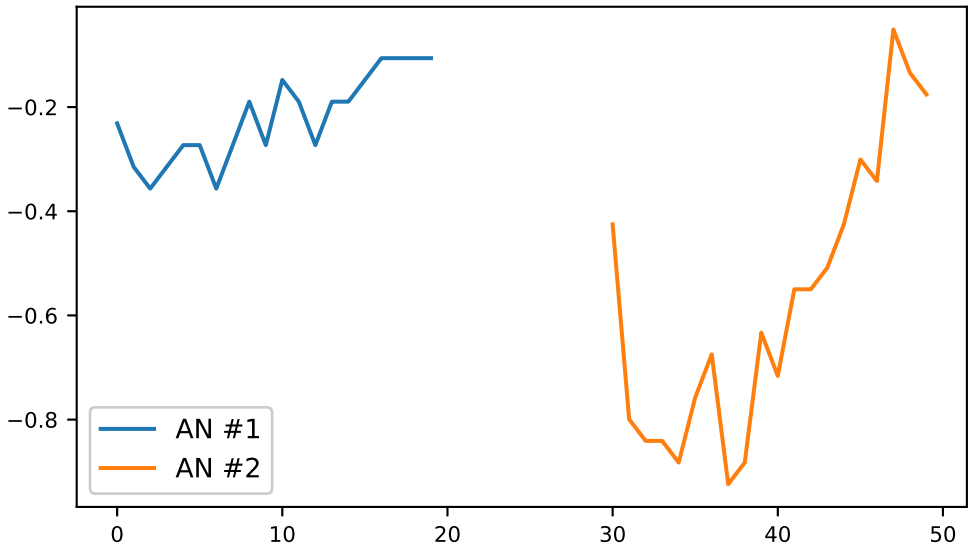

Figura 2.9: Janelas de tamanho 20 períodos e EMA de 5 períodos normalizadas por Normalização Adaptativa: AN #1 para o período de baixa volatilidade entre 9:50 e 10:10 e AN #2 para o período de alta volatilidade entre 11:00 e 11:20 do dia 1º de Setembro de 2016

2.4.4 Desnormalização e Destransformação dos dados

Os dados transformados e normalizados resultantes do processo de Normalização Adaptativa podem ser utilizados como entrada para um método de Aprendizagem de Máquinas, como Redes Neurais. Após previsões serem feitas, deve ocorrer uma desnormalização e destransformação dos dados com o objetivo de mapear as saídas obtidas aos valores originais da série temporal.

Primeiramente deve-se desnormalizar min-max. Dado um atributo A, o processo de desnormalização min-max converte um valor a' no intervalo $[low, high]$ a um valor a de A ao computar segundo a equação (2.11).

$$a = \frac{a' - low}{high - low} * (\max A - \min A) + \min A \tag{2.11}$$

Após a desnormalização ser realizada, é necessário realizar a destransformação, que irá converter os valores para valores similares a série temporal original. Para isso, basta multiplicar os valores desnormalizados pelos $S^{(k)}[i]$ corretos, ou seja, os que foram utilizados como denominador para cada janela agora serão usados para destransformar a série aos valores originais.

2.5 Modelo Auto-regressivo Integrado de Médias Móveis (*ARIMA*)

ARIMA é derivada de uma composição de processos de modelagem autorregressivos e de médias móveis, com a possibilidade de adição de um processo de diferenciação preliminar (SALLES *et al.*, 2017; BOX *et al.*, 2013). Modelos Autorregressivos (AR) assumem que um valor x_i de uma série temporal pode ser descrito como um função de seus p valores passados, $x_{i-1}, x_{i-2}, \ldots, x_{i-p}$. Um modelo AR de ordem p, AR(p), é definido na equação (2.12), onde x_t é estacionária, $\phi_1, \phi_2, \ldots, \phi_p$ são constantes com $\phi_p \neq 0$, e ω_t é uma série Gaussiana de ruído branco (*white noise*) com média zero e variância σ_ω^2 (BOX *et al.*, 2013).

$$x_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + \cdots + \phi_p x_{t-p} + \omega_t \tag{2.12}$$

Em contrapartida, o modelo de média móvel (MA) supõe que um valor x_i de uma série temporal pode ser descrito como uma função de seus q valores de ruído branco anteriores $\omega_{i-1}, \omega_{i-2}, \ldots, \omega_{i-q}$. O modelo de Média Móvel de ordem q, MA(q), é definido na equação (2.13), onde q é o número de atrasos das médias móveis e $\theta_1, \theta_2, \ldots, \theta_q$ ($\theta_q \neq 0$) são parâmetros (SHUMWAY e STOFFER, 2011). O modelo MA se difere do AR também por ser estacionário independente dos valores assumidos pelos parâmetros $\theta_1, \theta_2, \ldots, \theta_q$ (SHUMWAY e STOFFER, 2011).

$$x_t = \omega_t + \theta_1 \omega_{t-1} + \theta_2 \omega_{t-2} + \cdots + \theta_q \omega_{t-q} \tag{2.13}$$

Esses modelos podem ser combinados, gerando o modelo Autorregressivo de Médias Móveis (ARMA). O modelo ARMA (p, q) é definido pela equação (2.14), onde $\phi(B)$ e $\theta(B)$ são os operadores de AR e MA, respectivamente (BOX *et al.*, 2013).

$$\begin{aligned} &\phi(B)x_t = \theta(B)\omega_t \\ &\phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p \\ &\theta(B) = 1 + \theta_1 B + \theta_2 B^2 + \cdots + \theta_q B^q \end{aligned} \tag{2.14}$$

Por fim, o modelo integrado ARMA, ou modelo ARIMA, pode ser formulado incluindo um processo de diferenciação anterior, portanto preparado para lidar com séries temporais não estacionárias. Então, uma série temporal x_t é definida como um modelo autorregressivo integrado de médias móveis de ordem (p, d, q) ou ARIMA (p, d, q), representado pela equação (2.15) (SHUMWAY e STOFFER, 2011).

$$\phi(B)(1-B)^d x_t = \theta(B)\omega_t \tag{2.15}$$

Muitas vezes uma série temporal pode ter comportamentos completamente aleatórios ao longo do tempo. Nesses casos, um modelo útil para estimar esse tipo de série temporal é o ARIMA(0,1,0), considerado um modelo *random walk with drift* (SHUMWAY e STOFFER, 2011), onde o valor de uma série temporal no tempo t é explicado pelo valor da série no tempo $t-1$, mais um movimento completamente aleatório, representado por ω_t, mais uma constante δ, chamada de *drift*.

Muitos artigos utilizam previsões feitas por modelos ARIMA para comparar resultados de previsões feitas com Redes Neurais. Diversos exemplos podem ser vistos na literatura, como comparações no mercado de previsão de vendas (YAO e TAN, 2000), em negócios ou aplicações industriais (HO *et al.*, 2002; GHIASSI *et al.*, 2005). Existem ainda muitos outros artigos que avaliam a acurácia de Redes Neurais através do ARIMA na *survey* acerca do estado da arte sobre previsão com Redes Neurais Artificiais (ZHANG *et al.*, 1998).

Alguns experimentos foram realizados e relatados em artigos que utilizam modelos ARIMA na construção de sistemas de redes neurais híbridos, ao se juntar os dois conceitos. Geralmente os modelos híbridos superaram ambos os modelos aplicados separadamente (KHASHEI e BIJARI, 2011). Mesmo que o ARIMA tenha sido parte do sistema híbrido de previsão de séries temporais, ele também pode ser utilizado como *benchmark* em um estudo comparativo para se ter certeza que a abordagem híbrida é satisfatória ou superior.

Para ajustar um modelo ARIMA geralmente deve-se identificar a ordem autorregressiva p, a ordem de diferenciação d e a ordem da média móvel q (BOX *et al.*, 2013). Escolher os parâmetros corretos para produzir um modelo ARIMA não é tarefa trivial, alguns algoritmos foram desenvolvidos para computar esses parâmetros (HYNDMAN e KHANDAKAR, 2008; HYNDMAN e ATHANASOPOULOS, 2014).

Para este trabalho, é utilizado o pacote para linguagem R *TSPred* (SALLES *et al.*, 2017), o qual se utiliza da função *auto.arima* do pacote para linguagem R *forecast* (HYNDMAN e KHANDAKAR, 2008). Essa função tem o objetivo de selecionar um modelo ARIMA otimizado, definido pelos valores dos seus parâmetros p, d e q, computados através de uma variação do algoritmo de Hyndman e Khandakar (HYNDMAN e KHANDAKAR, 2008; HYNDMAN e ATHANASOPOULOS, 2014). Os modelos, criados por esse algoritmo, gerando cada previsão, são utilizados como *baseline* de comparação com os resultados das previsões feitas por Rede Neural, com os diversos tipos de normalização utilizadas nesse trabalho.

Importante ressaltar que as previsões com ARIMA não funcionam da mesma forma que Redes Neurais e outros métodos de aprendizagem de máquinas. Não se

pode dividir em treino e teste, nem é preciso normalizar os dados. É necessário usar toda a sequência de valores anteriores ao que se deseja prever como entrada do modelo para se retirar a saída, a qual será a previsão. Deve-se realizar o mesmo procedimento para cada elemento no conjunto de testes, para assim se poder comparar com a Rede Neural, no caso desse trabalho. Na Tabela 2.4 está exemplificado uma previsão por ARIMA dos 10 últimos valores da série temporal representada na Figura 2.2, os quais são considerados como conjunto de testes nesse caso.

Tabela 2.4: Previsão com ARIMA(2,1,1) dos 10 últimos minutos de mercado (17:51 às 18:00) do dia 1º de Setembro de 2016

t(m)	real	previsto
17:51	3281.5	3281.175036491769
17:52	3281.5	3281.9353046473047
17:53	3282.0	3281.642931430105
17:54	3281.5	3281.5111627167717
17:55	3282.0	3282.103667913121
17:56	3282.0	3281.5056737452574
17:57	3281.5	3282.134131472778
17:58	3283.0	3281.967987770419
17:59	3282.0	3281.6624224371476
18:00	3286.0	3282.964972403989
		RMSE: 1.071

Capítulo 3

Proposta

Neste capítulo é apresentado o problema existente na Normalização Adaptativa para séries com média móvel nula ao longo do tempo e duas propostas que visam solucionar o mesmo. A primeira alternativa é realizar uma adição de constante ao numerador e denominador no cálculo da AN, intitulada Normalização Adaptativa Compensada. Já a segunda, utiliza a subtração em vez da divisão neste cálculo, chamada de Normalização Adaptativa por Subtração.

3.1 O problema da AN para séries com média móvel nula ao longo do tempo

A AN possui uma limitação ao permitir que se ocorra divisão por zero. Isso acaba gerando valores indefinidos em casos específicos cuja média móvel para a série temporal, a qual seria usada como denominador na divisão da equação (2.10) para gerar uma janela deslizante desconectada, teria um valor nulo. A fase da AN de remoção de *outliers* eliminaria as janelas nessas condições. Portanto, a normalização ficaria prejudicada, já que determinadas janelas seriam completamente indefinidas e, caso se desejasse realizar previsões, certas faixas de valores seriam impossíveis de prever devido à falta de informação causadas por essas janelas "perdidas".

Séries que possuam médias móveis nulas somente ocorrem quando a série possui valores negativos e positivos ao longo do tempo, permitindo assim que ambos se anulem ao se calcular a média. Em casos específicos, como em séries com os valores de fechamento no mercado de ações ou de criptomoedas, médias móveis nulas nunca ocorrerão, já que não existem valores negativos capazes de anular os positivos existentes nessas séries temporais. Desta forma, a média móvel sempre será positiva, não permitindo que se ocorra divisão por zero. Porém, em séries como a senoidal e cossenoidal, é possível que ocorra média móvel com valor zero em determinados pontos da série, demonstrado na Figura 3.1, onde ocorrem os cruzamentos na linha

pontilhada.

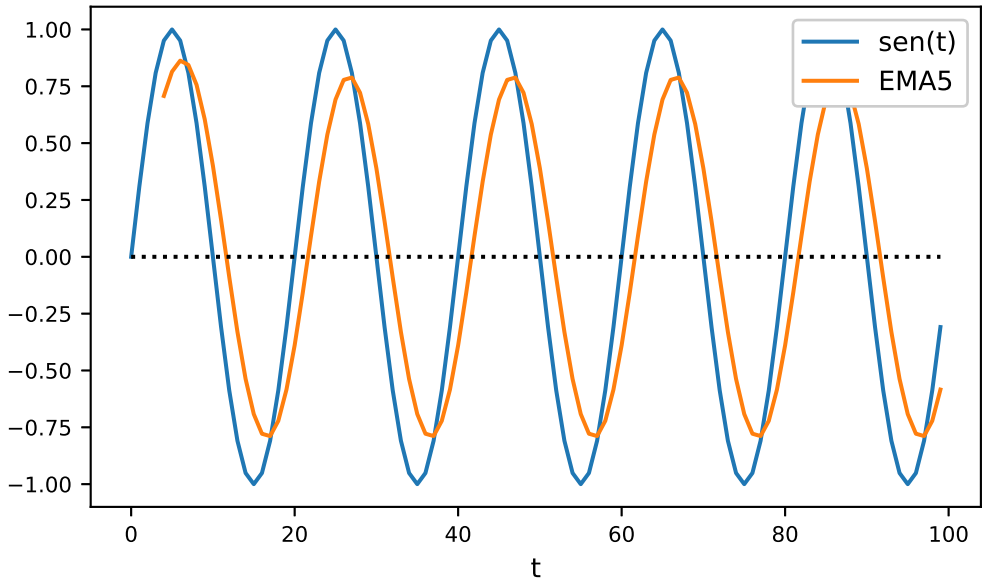

Figura 3.1: Exemplo de função senoidal e sua média móvel exponencial de ordem 5 (EMA5), possuindo valores nulos ao cruzar a linha pontilhada

Um outro exemplo de série passível de sofrer esse problema seria uma que representasse as variações percentuais de uma ação ou do câmbio, já que tais podem ocorrer tanto positivamente quanto negativamente. De forma a ilustrar como ocorre a aparição de valores nulos nas médias móveis, considera-se a série hipotética S, representada em (3.1). Nesse caso, utiliza-se a Média Móvel Simples (SMA), com fator de inércia $k = 4$ ($S_s^{(4)}$). Portanto, é realizada uma "varredura"pela série em agrupamentos de tamanho 4, de modo a calcular suas médias móveis.

$$
\begin{aligned}
S &= 2,-5,6,-2,3,0,-1,-4,6 \\
&= \underbrace{2,-5,6,-2}_{S_s^{(4)}=2-5+6-2=1},3,0,-1,-4,6 \\
&= 2,\ \underbrace{-5,6,-2,3}_{S_s^{(4)}=-5+6-2+3+0=2},0,-1,-4,6 \\
&= 2,-5,\ \underbrace{6,-2,3,0}_{S_s^{(4)}=6-2+3+0=7},-1,-4,6 \\
&= 2,-5,6,\ \underbrace{\mathbf{-2,3,0,-1}}_{\boldsymbol{S_s^{(4)}=-2+3+0-1=0}},-4,6 \\
&= 2,-5,6,-2,\ \underbrace{3,0,-1,-4}_{S_s^{(4)}=3+0-1-4=-2},6 \\
&= 2,-5,6,-2,3,\ \underbrace{0,-1,-4,6}_{S_s^{(4)}=0-1-4+6=1}
\end{aligned}
\tag{3.1}
$$

É possível verificar que o agrupamento em negrito possui um resultado nulo, ou seja, irá gerar um valor indefinido durante a realização da AN, pois será o denominador da equação (2.10) para alguma janela. Se for considerada uma janela de tamanho $w = 2$, a AN geraria novas janelas como no exemplo (3.2).

$$
\begin{aligned}
AN1 &= \frac{[3,0]}{S_s^{(4)}=1} = [3,0] \\
AN2 &= \frac{[0,-1]}{S_s^{(4)}=2} = [0,-1/2] \\
AN3 &= \frac{[-1,-4]}{S_s^{(4)}=7} = [-1/7,-4/7] \\
\boldsymbol{AN4} &= \frac{[-4,6]}{S_s^{(4)}=0} = \underbrace{[indefinido, indefinido]}_{eliminada}
\end{aligned}
\tag{3.2}
$$

A janela AN4, em destaque, obtém valores indefinidos e não poderá ser utilizada, sendo eliminada durante o processo de remoção de *outliers* da AN original.

Visando solucionar essa questão, são propostas duas alterações à Normalização Adaptativa. As soluções propostas a seguir: Normalização Adaptativa Compensada e Normalização Adaptativa por Subtração.

3.2 Normalização Adaptativa Compensada

A Normalização Adaptativa Compensada (ANC) adiciona uma mesma constante ao numerador e denominador da equação (2.10), ou seja, é somada essa constante para cada elemento da subsequência original no numerador, assim como à média móvel no denominador. Visando simplificar os cálculos, utiliza-se o menor número inteiro positivo como constante. A equação, então, fica como descrita em (3.3).

$$r[i] = \frac{S[i, \ldots, (i+w-1)] + 1}{S^{(k)}[i] + 1}, \forall i / w \leq i \leq (n-w+1) \tag{3.3}$$

Ao se considerar a mesma série S, representada em (3.1), pode-se comparar como a ANC se comporta em (3.4).

$$\begin{aligned}
ANC1 &= \frac{[3,0]}{S_s^{(4)} = 1} = [(3+1)/(1+1), (0+1)/(1+1)] = [2, 1/2] \\
ANC2 &= \frac{[0,-1]}{S_s^{(4)} = 2} = [(0+1)/(2+1), (-1+1)/(2+1)] = [1/3, 0] \\
ANC3 &= \frac{[-1,-4]}{S_s^{(4)} = 7} = [(-1+1)/(7+1), (-4+1)/(7+1)] = [0, -3/8] \\
\boldsymbol{ANC4} &= \frac{[-4,6]}{S_s^{(4)} = 0} = [(-4+1)/(0+1), (6+1)/(0+1)] = \boldsymbol{[-3, 7]}
\end{aligned} \tag{3.4}$$

Observa-se que, a ANC4 em destaque não possui mais valores indefinidos, permitindo, então, que essa janela possa ser utilizada. Também, é possível ver que os valores obtidos nas normalizações são bem diferentes dos obtidos com a AN original, já que a ordem de grandeza da constante é a mesma dos valores originais, portanto, influenciando significativamente nos cálculos. Se a ordem de grandeza dos valores presentes na série original forem próximas da constante de valor 1 - como acontece nesse caso -, a ANC irá influenciar bastante no resultado da normalização em relação à AN original. Com a utilização da normalização Min-Max posterior, as janelas geradas serão bem semelhantes e cumprirão com o objetivo da AN original: representar a volatilidade em cada janela gerada.

Após a ANC ser normalizada por Min-Max, ou seja, a ANC estar realizada por completo, vê-se que, na ANC, a normalização fica muito próxima da AN original quando os valores originais são de ordem de grandeza bem maior que a constante inserida. Nesse caso, mesmo antes de se realizar a normalização Min-Max, os valores representados nas janelas já seriam muito próximos, pois a influência de somar um valor de ordem de grandeza muito pequena a outro com uma ordem de grandeza muito maior é desprezível.

3.3 Normalização Adaptativa por Subtração

A Normalização Adaptativa por Subtração (ANS) realiza uma subtração entre o numerador e denominador da equação (2.10). A equação, então, fica como descrita em (3.5).

$$r[i] = S[i, \ldots, (i + w - 1)] - S^{(k)}[i], \forall i / w \leq i \leq (n - w + 1) \tag{3.5}$$

Ao se considerar a mesma série S, representada em (3.1), pode-se comparar como a ANS se comporta em (3.6).

$$\begin{aligned} ANS1 &= [3, 0] - (S_s^{(4)} = 1) = [(3 - 1), (0 - 1)] = [2, -1] \\ ANS2 &= [0, -1] - (S_s^{(4)} = 2) = [(0 - 2), (-1 - 2)] = [-2, -4] \\ ANS3 &= [-1, -4] - (S_s^{(4)} = 7) = [(-1 - 7), (-4 - 7)] = [-8, -11] \\ \boldsymbol{ANS4} &= [-4, 6] - (S_s^{(4)} = 0) = [(-4 - 0), (6 - 0)] = \boldsymbol{[-4, 6]} \end{aligned} \tag{3.6}$$

Observa-se que, a ANS4 em destaque também não possui mais valores indefinidos, permitindo, então, que essa janela possa ser utilizada. Também, é possível ver que os valores obtidos nas normalizações são bem diferentes dos obtidos com a AN original. É um comportamento esperado, já que se trata de uma operação matemática diferente, a troca de uma operação de divisão por uma de subtração. Assim como na ANC, com a utilização da normalização Min-Max posterior, as janelas geradas serão bem semelhantes e cumprirão com o objetivo da AN original: representar a volatilidade em cada janela gerada.

Capítulo 4

Avaliação Experimental

Neste capítulo são realizadas experimentações com os diversos tipos de Normalização explicadas no capítulo de Fundamentação Teórica, assim como das duas novas propostas provenientes de adaptações na Normalização Adaptativa, cuja é foco principal deste trabalho. Como forma de validar a aplicabilidade das normalizações em diversos tipos de séries temporais, são utilizados diversos *datasets* de diferentes áreas (mercado financeiro, cripto-moedas, volume de chuvas, etc.), assim como periodicidades (por minuto, por ano, por mês, etc.).

Este capítulo se organiza da seguinte forma: Primeiramente descrevem-se os experimentos a serem realizados; depois, a metodologia utilizada para validar as propostas, descrevendo os *datasets* utilizados, configurações do experimento e métricas; tecnologia utilizada; e resultados e análise dos experimentos.

4.1 Objetivos dos experimentos

Os experimentos tem como objetivo incrementar os que já foram realizados no artigo que introduziu o conceito de Normalização Adaptativa (OGASAWARA *et al.*, 2010). Lá, foram aplicadas 5 diferentes normalizações (Min-Max, Decimal, Z-score, *Sliding Windows*, Normalização Adaptativa) como parte do pré-processamento para aplicação em uma Rede Neural, sendo todas comparadas com o método AR. Novamente, são testadas as 5 normalizações citadas anteriormente, assim como as duas novas normalizações propostas neste trabalho, todos comparadas com o *baseline* ARIMA.

Importante ressaltar que os resultados obtidos para o ARIMA nesse trabalho são bem melhores que os obtidos para o AR nos exemplos de (OGASAWARA *et al.*, 2010), já que nesse trabalho os parâmetros são aplicados ao ARIMA automaticamente a partir de uma variação do Algoritmo de Hyndman e Khandakar (SALLES *et al.*, 2017; HYNDMAN e KHANDAKAR, 2008; HYNDMAN e ATHANASOPOULOS, 2014), enquanto para o trabalho de Ogasawara o parâmetro é fixo. Há também

o diferencial de se testar a Normalização Adaptativa em séries temporais de período curto, na ordem dos minutos.

Enumeram-se, então, os seguintes experimentos e seus objetivos:

- Experimento I: Comparação entre os métodos de normalização para o *dataset* de contratos MINI Dólar por minuto.
- Experimento II: Comparação entre os métodos de normalização para o *dataset* de bitcoin por minuto.
- Experimento III: Comparação entre os métodos de normalização para o *dataset* de volume de chuvas anuais (em mm) na cidade de Fortaleza, Ceará.
- Experimento IV: Comparação entre os métodos de normalização para o *dataset* de vazões médias mensais (em m^3/s) da usina hidrelétrica de FURNAS.
- Experimento V: Comparação entre os métodos de normalização para o *dataset* de variações percentuais de contratos MINI Dólar por minuto.

4.2 Metodologia

Aqui se descreve a Metodologia utilizada para avaliar as propostas, incluindo descrição dos *datasets* usados; configuração dos experimentos, descrição dos algoritmos de treinamento, arquitetura da Rede Neural, assim como parâmetros utilizados; e descrição das métricas aplicadas.

4.2.1 *Datasets*

Para esse trabalho foram utilizados 5 *datasets* que representam diferentes séries temporais de diferentes tamanhos. Todos os dados foram divididos em 80% para treino e 20% para testes. Uma descrição detalhada dos *datasets* utilizados se encontra a seguir:

- *Dataset* I: Representa os valores de fechamento por minuto dos contratos futuros de MINI Dólar para Setembro de 2016 (código na BOVESPA: WDOU16). Possui 10802 registros no total, os quais são posteriormente divididos entre treino e teste, respeitando a ordem cronológica por se tratar de uma série temporal. Os registros são referentes a cada minuto enquanto o mercado está aberto, ou seja, entre 9:00 e 18:00, durante os dias úteis. Portanto, apenas de segunda-feira à sexta-feira, excluindo-se feriados na cidade de São Paulo. Os dados foram extraídos da BM&FBOVESPA [1].

[1] BM&FBOVESPA. 2018. URL: `http://www.bmfbovespa.com.br/pt_br/`

- *Dataset* II: Representa os valores de fechamento por minuto da cripto-moeda Bitcoin para Dezembro de 2017 com valores expressos em dólar. Possui 44354 registros no total, os quais são posteriormente divididos entre treino e teste, respeitando a ordem cronológica por se tratar de uma série temporal. Os registros são referentes a cada minuto durante as 24h dos dias no mês inteiro de dezembro de 2017. Para o Bitcoin não existe abertura e fechamento de mercado, é como se ele estivesse sempre aberto, mesmo nos feriados e finais de semana, por isso existem muito mais registros para um mês em relação a papéis da BOVESPA. Os dados foram extraídos da API da GDAX [2], uma empresa que fornece ferramentas para *trading* de criptomoedas.

- *Dataset* III: Representa a quantidade de chuvas, medidas em milímetros, ocorridas por ano na cidade de Fortaleza, Ceará, entre 1850 e 1979, totalizando 130 registros. Os dados foram retirados de uma coleção de mais de 300 *datasets* de séries temporais, chamada *Hipel-McLeod Time Series Datasets Collection* (HIPEL e MCLEOD, 1994).

- *Dataset* IV: Representa o volume de vazões médias mensais, medidas em metros cúbicos por segundo, ocorridas na usina hidrelétrica de FURNAS, entre janeiro de 1931 e dezembro de 1978, totalizando 576 registros. Esses dados também foram retirados da coleção *Hipel-McLeod Time Series Datasets Collection* (HIPEL e MCLEOD, 1994).

- *Dataset* V: Representa as variações percentuais dos valores de fechamento por minuto dos contratos futuros de MINI Dólar para Setembro de 2016 (código na BOVESPA: WDOU16). As variações foram computadas a partir do *Dataset* I. Possui 10801 registros no total, um a menos que o *Dataset* I, já que não é possível calcular a variação do primeiro fechamento, os quais são posteriormente divididos entre treino e teste, respeitando a ordem cronológica por se tratar de uma série temporal. Os registros são referentes a cada minuto enquanto o mercado está aberto, ou seja, entre 9:00 e 18:00, durante os dias úteis, com exceção do primeiro minuto do primeiro dia, o qual não foi possível calcular a variação em relação ao dia anterior por falta de dados. Portanto, apenas de segunda-feira à sexta-feira, excluindo-se feriados na cidade de São Paulo.

4.2.2 Configuração dos experimentos

Como citado anteriormente, os dados são divididos em 80% para treino e 20% para teste. Porém, dos 80% dos dados no conjunto de treino, os últimos 10% desses

[2]GDAX. 2018. URL: `https://www.gdax.com/`

dados são utilizados como conjunto de validação, dessa forma cada *dataset* fica dividido em 70% para treino, 10% para validação e 20% para teste, em um treinamento com 1000 épocas. O conjunto de validação é utilizado para se verificar melhoras no erro durante o treinamento da rede após cada época, permitindo assim utilizar determinadas técnicas durante a fase de treinamento de modo a otimizar a rede.

Afim de se manter uma reprodutibilidade dos experimentos são fixadas 10 sementes responsáveis por gerar números aleatórios (com os seguintes valores: $4395, 3129, 277, 9871, 5183, 6082, 810, 6979, 2654, 5765$, todos números entre 1 e 10000 gerados de forma aleatória), além de fixar o programa para utilizar somente uma *thread*, segundo as recomendações do Keras (biblioteca contendo algoritmos de Redes Neurais utilizada) para reprodutibilidade. Essas 10 sementes são utilizadas em cada vez que é executado algum treinamento de rede para cada experimento, portanto gerando 10 RMSE diferentes para cada configuração, dessa forma podemos obter uma média dos RMSE obtidos, diminuindo a incerteza dos resultados, ou até mesmo plotar via *boxplot*. Assim, pode-se obter um resultado mais seguro de que a Rede Neural está convergindo para os melhores resultados de acordo com a sua configuração ou com o método de normalização aplicado, e, enfim, se realizar a análise comparativa desejada.

Quando se treina um modelo é necessário evitar *overfitting*, ou seja, um sobreajuste ao conjunto de dados de treino, porém ineficaz na previsão de dados novos do teste. Existem técnicas que ajudam a evitar isso em Redes Neurais, como o uso de *Dropout* (SRIVASTAVA *et al.*, 2014), que remove aleatoriamente neurônios da Rede para prevenir que eles se coadaptem demais, ou encerrar o treinamento no momento que o erro no conjunto de dados de validação comece a piorar (*early stopping*) (CARUANA *et al.*, 2000; PRECHELT, 1998), assim como métodos que introduzem penalidades aos pesos como as regularizações L1 e L2 ou *soft weight sharing* (NOWLAN e HINTON, 1992). Essas regularizações são úteis para muitos modelos e é utilizada neste trabalho a fim de melhorar os resultados e evitar *overfitting*, pois adicionam um termo de regularização, prevenindo que os coeficientes se adéquem perfeitamente. A diferença entre as regularizações L1 e L2 é apenas que a primeira utiliza a soma dos pesos, enquanto a segunda usa o quadrado da soma dos pesos como termo de regularização. Nos experimentos realizados nesse trabalho, a única técnica não utilizada é a *early stopping*, pois pretende-se oferecer as mesmas condições de treinamento para todas as normalizações e *datasets* diferentes, ou seja, o mesmo número de épocas fixadas para que os modelos sejam treinados.

Como entrada da Rede Neural é utilizada a quantidade de neurônios equivalente ao tamanho da janela w. Essa janela representa a quantidade de períodos anteriores ao que se deseja prever, ou seja, uma janela de tamanho 30, por exemplo, representaria os 30 períodos anteriores ao que se deseja prever, portanto, nesse caso,

seriam 30 *features* sendo representados como 30 neurônios na camada de entrada. Cabe ressaltar que essa metodologia se difere da aplicada em (OGASAWARA *et al.*, 2010), onde foi fixada a quantidade de neurônios em 3, independente do tamanho da janela w.

O tamanho da janela w ideal, ou seja, que terá erros menores, varia de acordo com o *dataset* utilizado, assim como a normalização. Já para camada oculta foi realizada uma pesquisa empírica ao se variar a quantidade de camadas ocultas entre 1 e 2 e a quantidade de neurônios em cada camada entre 3, 6, 9 e 12. E a camada de saída, por se tratar de um problema de regressão, possui apenas um neurônio, o qual irá apresentar o valor previsto pela rede neural.

Utiliza-se a mesma função de ativação *tanh* (tangente hiperbólica), utilizada em (OGASAWARA *et al.*, 2010), representada na Figura 4.1, para todas as camadas. Tal função de ativação possui como uma das características uma convergência mais rápida que seu par não-simétrico, a sigmoide logística, a qual possui um intervalo entre 0 e 1, ou seja, limitando as saídas dos neurônios para essa faixa, introduzindo assim um viés sistemático a esses neurônios (HAYKIN *et al.*, 2009). Para uma convergência mais rápida é necessário que as médias das saídas dos neurônios estejam próximas de zero, o que ocorre mais facilmente com funções de ativação antissimétricas, como a *tanh*, que possui intervalo entre -1 e 1. Por fim, quanto maior a conectividade da Rede Neural, o aprendizado por *backpropagation* atinge convergência mais rápido com funções antissimétricas do que não-simétricas (LECUN *et al.*, 1998, 1991).

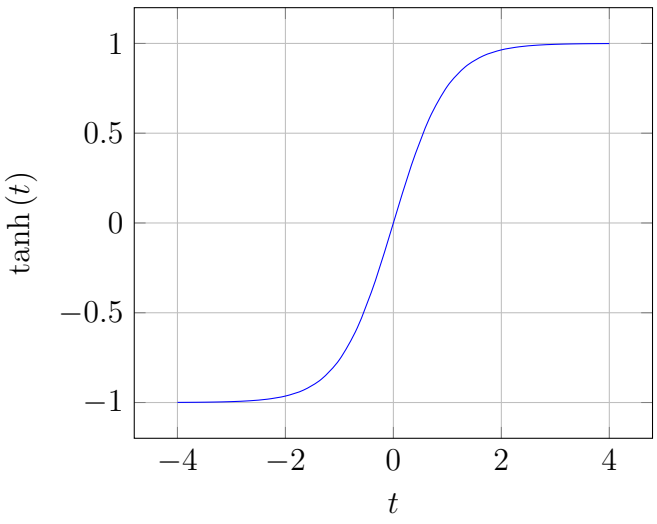

Figura 4.1: Função de Ativação *tanh*

A fim de se evitar *overfitting*, uma camada de Dropout é utilizada entre a camada de entrada e a camada oculta, com uma taxa de Dropout de 0.25, ou seja, por exemplo, para cada 4 neurônios da camada de entrada, 1 será eliminado de forma

aleatória em cada época. Também visando evitar *overfitting* e uma melhora no erro, é utilizada a regularização L2 com taxa 0.01, a regularização L1 foi preterida pois trouxe resultados piores em relação a L2 nos testes realizados. As taxas aqui utilizadas foram as que obtiveram melhores resultados nos testes com o *Dataset* I, as quais foram fixadas e utilizadas nos outros experimentos, de forma a manter o mesmo padrão.

A rede foi treinada utilizando o algoritmo de otimização *Adam* (KINGMA e BA, 2014) com *batches* de tamanho 64. Utiliza-se esse algoritmo pois ele é eficiente, atingindo bons resultados de forma rápida, se tornando bastante popular e recomendado como padrão a se utilizar dentre outras opções de algoritmos de otimização do tipo Gradiente Descendente (RUDER, 2016). Além disso, todos os pesos da rede são inicializados utilizando a regra uniforme de Glorot (GLOROT e BENGIO, 2010), a qual representa uma distribuição uniforme entre os limites $[-limit, limit]$ representados pela equação (4.1), onde fan_{in} é a quantidade de unidades de entrada no tensor de pesos, e fan_{out} a quantidade de unidades de saída no tensor de pesos:

$$limit = \sqrt{\frac{6}{(fan_{in} + fan_{out})}} \tag{4.1}$$

4.2.3 Métricas

Para todos os experimentos são utilizados o Erro Quadrático Médio (*Mean Squared Error* ou MSE), exibido na equação (4.2), como função de custo durante o treinamento. Porém, nesse trabalho é utilizada a Raiz do Erro Quadrático Médio (*Root Mean Squared Error* ou RMSE), que nada mais é que a raiz quadrada do MSE, demonstrado na equação (4.3), como métrica principal para avaliar a qualidade dos resultados. O RMSE possui a vantagem de ser mais facilmente interpretado, já que possui a mesma escala de unidades que os valores originais introduzidos no modelo, sendo assim uma estatística mais direta. Ambas as técnicas visam medir a diferença entre os valores previstos ($\hat{Y}_i$) pelo modelo e os valores reais presentes no conjunto de testes (Y_i). Importante ressaltar que essas métricas são dependentes de escala, ou seja, dependem da escala dos valores de cada *dataset*, portanto somente é válido comparar em valores absolutos os RMSE ou MSE para um mesmo *dataset*. Para se comparar o desempenho entre diferentes *datasets* das técnicas de normalização utilizadas, uma forma seria usar a diferença percentual entre as técnicas para se saber o quanto uma desempenhou-se melhor que a outra, como aplicado no artigo (OGASAWARA *et al.*, 2010), e então comparar essas diferenças de desempenho entre os diferentes *datasets*.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{4.2}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2} \tag{4.3}$$

Como dito anteriormente, os *datasets* foram divididos em conjuntos de treino, validação e teste, portanto, todas as referências aos RMSE calculados são relativos apenas ao conjunto de testes. O conjunto de validação é utilizado apenas para otimização do treinamento.

4.2.4 Tecnologia utilizada

Todos os experimentos foram realizados utilizando a linguagem de programação Python 3.6, utilizando-se de diversas bibliotecas como Numpy, Pandas e Math, assim como a biblioteca Keras [3] com backend Tensorflow (ABADI *et al.*, 2015). Para a construção dos gráficos foi utilizada a biblioteca Matplotlib (HUNTER, 2007).

4.3 Resultados e análise

Tendo sido apresentadas as configurações e tecnologias utilizadas para a realização dos experimentos, a seguir são demonstrados os resultados obtidos e analisados pontos relevantes que foram observados.

4.3.1 Experimento I

Neste experimento utiliza-se o *Dataset* I - contendo os valores de fechamento por minuto dos contratos futuros de MINI Dólar para Setembro de 2016, representado na Figura 4.2 - para treinamento e avaliação dos resultados. Compara-se os RMSE obtidos para cada uma das normalizações utilizadas como parte do pré-processamento no treinamento da rede, assim como o RMSE obtido com o ARIMA, o qual é usado como *baseline* neste trabalho.

[3]Keras. 2018. URL: `https://keras.io/`

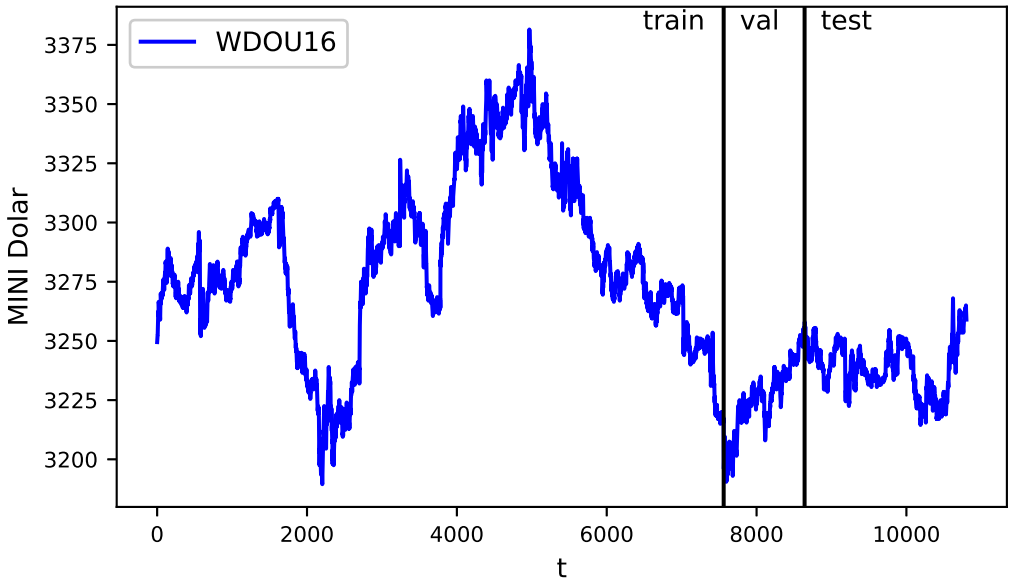

Figura 4.2: *Dataset* I, com cortes representando os conjuntos de treino, validação e teste

Afim de se encontrar uma boa configuração de Rede Neural foi realizada uma busca empírica. Variou-se ela entre 1 e 2 camadas ocultas e entre 3, 6, 9 e 12 neurônios por camada oculta, utilizando a AN original como método de normalização durante o pré-processamento, fixando o k em 7 e o w em 8, e utilizando a Média Móvel Exponencial (EMA), de acordo com o primeiro exemplo demonstrado no artigo (OGASAWARA *et al.*, 2010). Os gráficos com os *boxplots* representando os 10 RMSE obtidos em cada configuração estão representados nas Figuras 4.3, para uma camada oculta, e 4.4, para duas camadas ocultas. A melhor configuração obtida foi com apenas uma camada oculta e 12 neurônios. Porém, não há uma grande influência nos resultados com a variação de camadas ou neurônios.

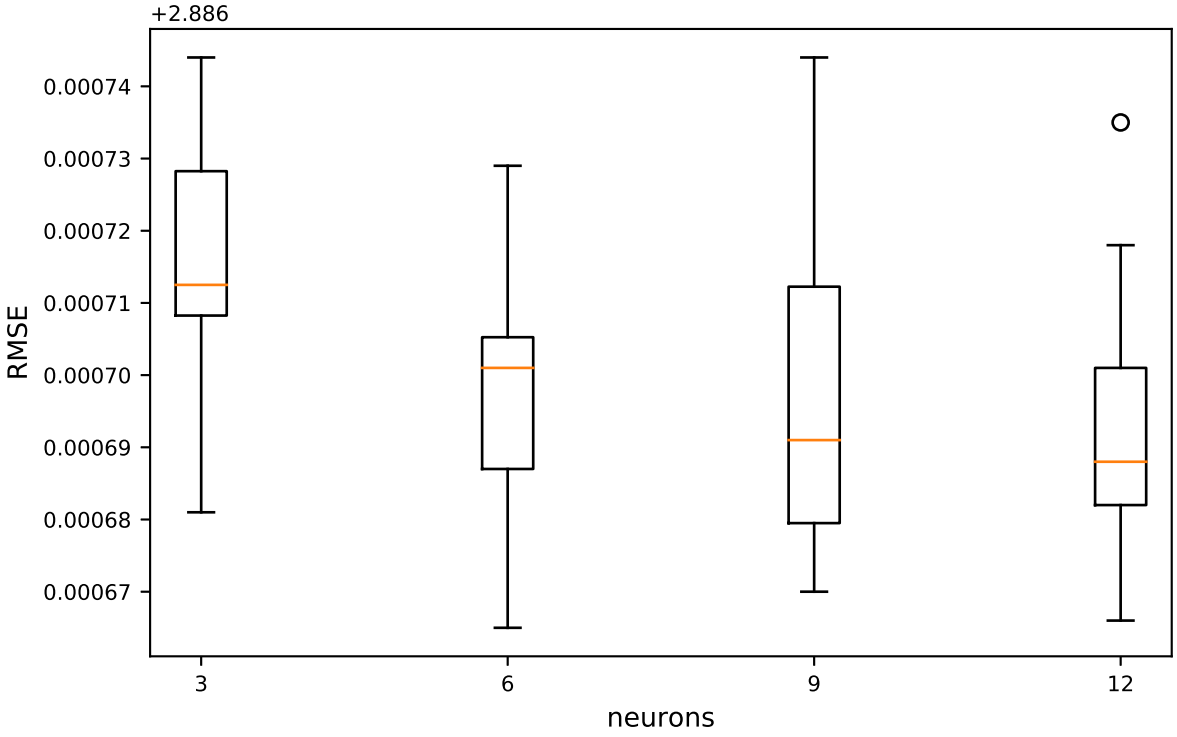

Figura 4.3: Rede Neural com uma camada oculta para *Dataset* I

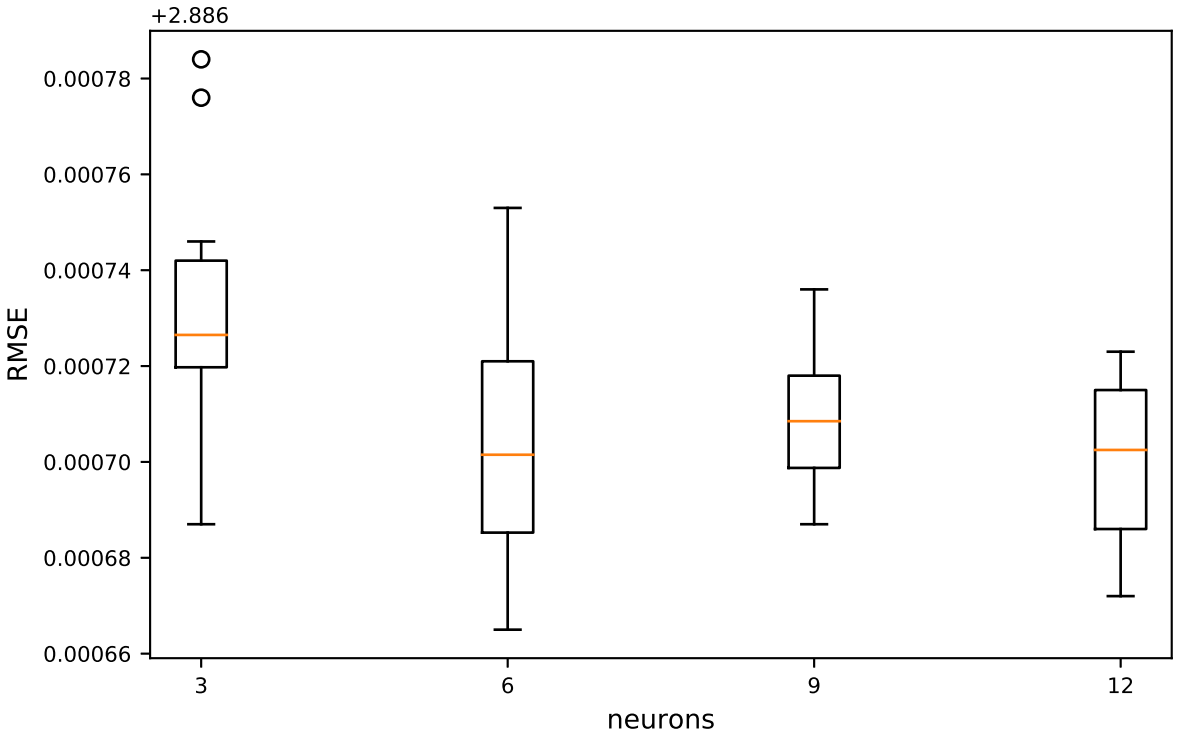

Figura 4.4: Rede Neural com duas camadas ocultas para *Dataset* I

Após encontrar a melhor configuração da estrutura da Rede Neural dentre as opções testadas, varia-se o fator de inércia k da MA entre 3, 8, 13, 18, 23 e 28. Não faz sentido k menor que 2, pois seria o mesmo que não haver MA nenhuma. O gráfico contendo as médias dos 10 RMSE obtidos para cada k pode ser visualizado na Figura 4.5. O melhor fator de inércia k encontrado dentre as opções testadas foi com o valor de 3. É possível observar que, quanto menor o k, menores os erros obtidos.

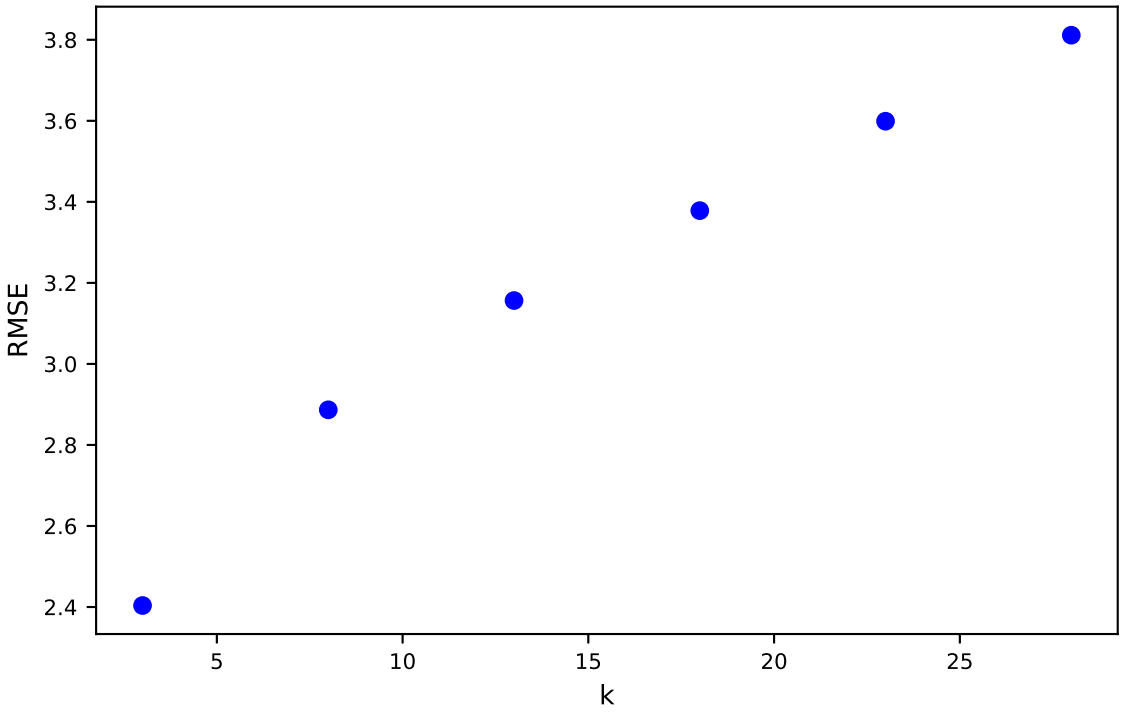

Figura 4.5: Desempenho da Rede Neural por fator de inércia da Média Móvel Exponencial para *Dataset* I

No seguinte passo, varia-se o tamanho w das janelas geradas entre 2, 7, 12, 17, 22 e 27. Não faz sentido w menor que 2, pois seria o mesmo que não haver janela nenhuma. Esse valor representa o número de entradas ou de neurônios na camada de entrada da Rede Neural. O gráfico contendo as médias dos 10 RMSE obtidos para cada w pode ser visualizado na Figura 4.6. O melhor tamanho para janela w encontrado dentre as opções testadas foi com o valor de 2. É possível observar que, quanto menor o w, menores os erros obtidos.

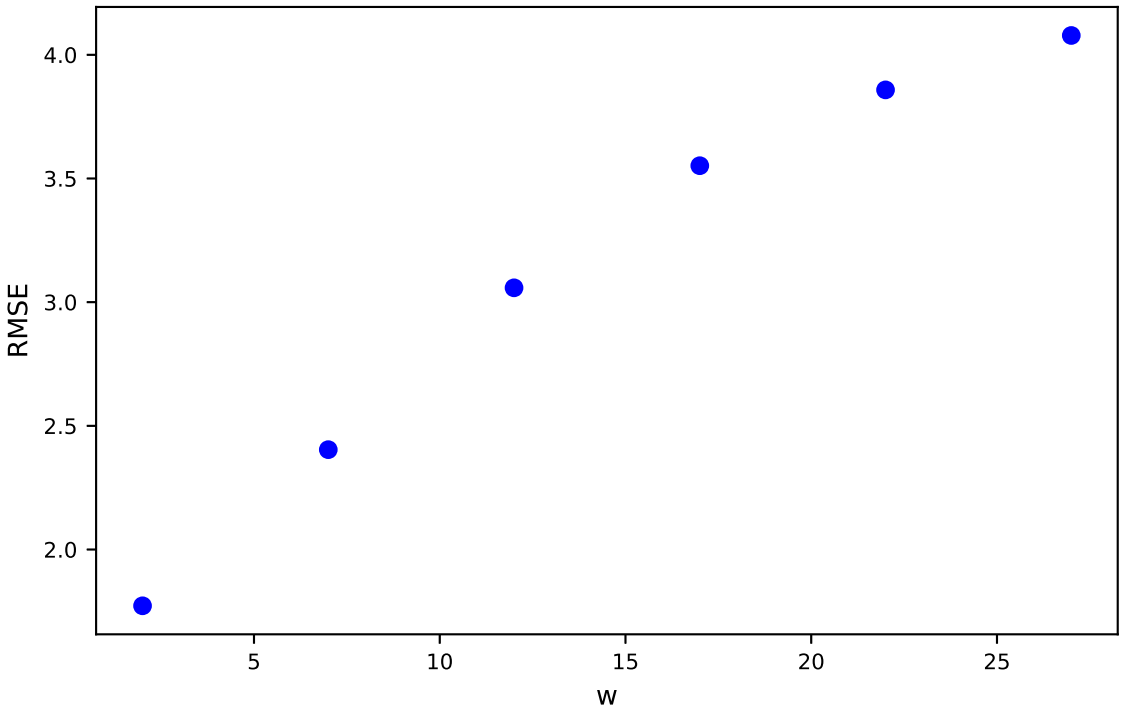

Figura 4.6: Desempenho da Rede Neural por tamanho de janela para *Dataset* I

Tendo sido encontrada a melhor configuração, ou seja, a que obteve o menor RMSE com a utilização da normalização AN original no pré-processamento, realizam-se experimentos com os mesmos parâmetros para cada normalização descrita nesse trabalho, visando comparar os resultados de forma justa. Essa abordagem é mais direta que a dada pela equação (**??**), pois compara-se logo os resultados através dos erros obtidos utilizando o conjunto de validação ou mesmo o conjunto de testes. Os RMSE obtidos para cada normalização, assim como o obtido com a utilização do ARIMA, estão na Tabela 4.1. Lembrando que os RMSE representados na tabela são a média dos 10 RMSE obtidos ao rodar os experimentos com as 10 sementes fixas anteriormente, com exceção do ARIMA, cujo resultado é único por *dataset*.

Tabela 4.1: Desempenho dos algoritmos utilizados para previsões no *Dataset* I

Algoritmo	RMSE
ARIMA	1.2658335
NN-MM	2.7341884
NN-DS	44.1148913
NN-ZS	8.3969501
NN-SW	1.3496186
NN-AN	1.7724663
NN-ANC	1.7724666
NN-ANS	**1.1713768**

O menor RMSE obtido foi de 1.1713768, com a ANS. Pode-se perceber que apenas esse método, ao ser aplicado junto à Rede Neural, superou o erro obtido com o *baseline* ARIMA. O NN-ANS (*Neural Network*-ANS) se mostrou 7,46% superior ao ARIMA, ou seja, obteve um RMSE 7,46% menor que o obtido com o *baseline*. Além disso, o método proposto ANS é superior em relação ao AN em 33,91% para esse *Dataset* com essas configurações.

Os outros ANs não conseguiram bater o SW em nenhum dos testes realizados com esse *Dataset*, diferente do que se pode observar nos exemplos mostrados no artigo (OGASAWARA *et al.*, 2010), em que o AN original sempre se mostrava superior tanto ao SW quanto ao ARIMA. Isso provavelmente se deve ao fato de que os períodos estão por minuto, enquanto no artigo são utilizados períodos maiores, como mensalmente ou trimestralmente, dessa forma a volatilidade é menor, o que talvez seja mais favorável à AN. Há também o fator de que são utilizados o número de neurônios como entrada igual ao tamanho da janela w, enquanto no artigo são fixados em três, independente do w. O diferencial do método AN ou ANC para o ANS é que a divisão de valores pelas médias móveis acaba por muitas vezes trazer resultados muito próximos de 1, normalmente variando entre 0.9 e 1.1, para então serem normalizados com o método Min-Max, já o ANS, por realizar uma subtração, gera um intervalo bem maior, podendo até trazer valores negativos, caso a média móvel seja superior ao primeiro elemento da janela observada. Desta forma, a normalização Min-Max traz valores razoavelmente diferentes entre a AN e a ANS, mesmo que a ideia por trás de ambas seja a mesma, mudando-se apenas uma parte da equação, como visto na Tabela **??**. Essas diferenças já são suficientes para melhorarem as previsões da Rede Neural.

Ressalta-se também a importância da remoção de *outliers* para esse caso, já que esse *Dataset* pode sofrer variações bruscas de um dia para outro, pois existem momentos em que o mercado não está aberto. Então, não se pode operar, e os valores

não variam nesse espaço de tempo. Esse tempo sem alteração pode gerar certos "saltos"de valores entre o fechamento e a abertura do dia seguinte, devido à uma demanda acumulada por vender ou comprar ativos. Normalmente, os períodos de abertura e fechamento de mercado são considerados períodos de maior instabilidade já que ocorre uma maior volatilidade e um volume de operações maior. Portanto, são períodos difíceis de prever e que acabam sendo eliminados dos conjuntos de treino e teste, graças a remoção de *outliers*.

Observa-se que, como no artigo (OGASAWARA *et al.*, 2010), os algoritmos NN-ANs, NN-SW e ARIMA são os que obtém melhores resultados. Tanto a AN quanto a ANC, como previsto anteriormente, devido à pequena diferença entre suas normalizações (Tabela **??**), obtiveram erros muito semelhantes. Portanto, a solução ANC promete resolver o problema com médias nulas da AN original, afetando sensivelmente, de forma quase imperceptível, os resultados que se obteriam com a AN. Dentre os restantes, o método NN-MM se sai melhor, provavelmente por que os dados de teste estão dentro do intervalo de mínimo e máximo obtidos do conjunto de treino, como pode ser visto na Figura 4.2, assim não ocorrendo perda de informações durante a normalização Min-Max. Como esperado, NN-ZS não consegue lidar bem com séries não-estacionárias, portanto seu RMSE se destaca negativamente.

Já o NN-DS se mostrou o pior de todos, apesar de que no artigo que apresenta a AN ele se sair razoavelmente bem nos exemplos, normalmente vindo logo atrás da NN-SW. Isso se deve ao fato de que a NN-DS não foi capaz de convergir com esses dados e essa configuração de rede. Isso provavelmente ocorre por que a normalização DS, ao ser aplicada nesse *Dataset* específico, gera valores com variações muito baixas, por exemplo, um valor próximo do mínimo 3200 ficaria 0.32, enquanto um valor próximo do máximo de 3375 ficaria 0.3375, ou seja, mesmo em extremos opostos nesse *Dataset*, como pode ser visto na Figura 4.2, essa normalização acaba por não diferenciar tanto os valores de entrada, dificultando a convergência da Rede Neural.

Concluindo, verifica-se que o método proposto ANS é o melhor método para o *Dataset* I, superando tanto o *baseline* como o AN original. Porém, é importante testá-lo em outros *Datasets* para validar a sua eficiência de forma generalizada, o que é feito nos experimentos seguintes.

4.3.2 Experimento II

Neste experimento utiliza-se o *Dataset* II - contendo os valores de fechamento por minuto da criptomoeda Bitcoin para Dezembro de 2017 com valores expressos em dólar, representado na Figura 4.7 - para treinamento e avaliação dos resultados. Compara-se os RMSE obtidos para cada uma das normalizações utilizadas como parte do pré-processamento no treinamento da rede, assim como o RMSE obtido

com o ARIMA, o qual é usado como *baseline* neste trabalho.

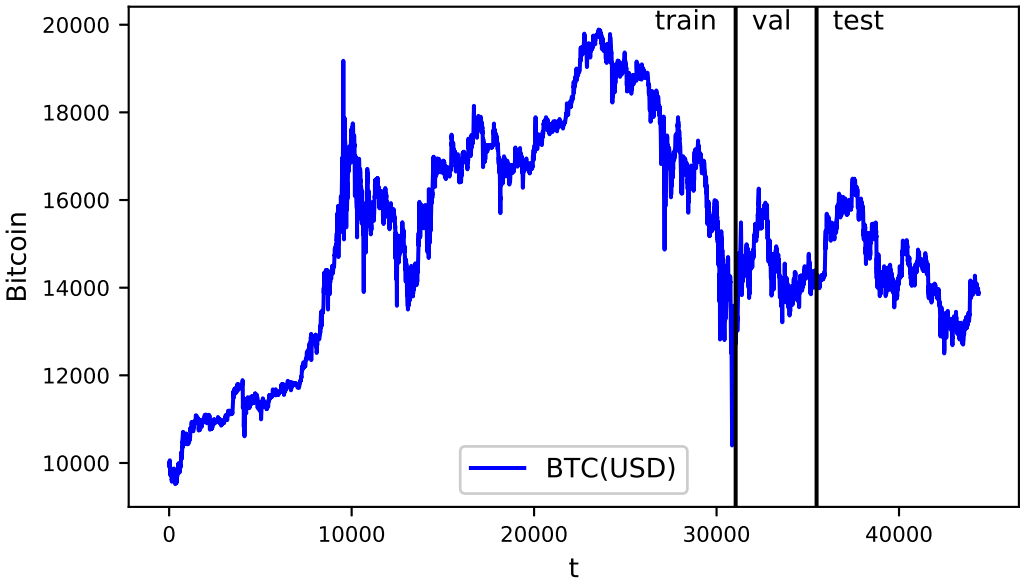

Figura 4.7: *Dataset* II, com cortes representando os conjuntos de treino, validação e teste

Assim como no experimento anterior, foi realizada uma busca empírica para encontrar uma boa configuração de Rede Neural. Variou-se ela entre 1 e 2 camadas ocultas e entre 3, 6, 9 e 12 neurônios por camada oculta, utilizando a AN original como método de normalização durante o pré-processamento, com o $k = 7$ e o $w = 8$, e utilizando a Média Móvel Exponencial (EMA), como no experimento anterior. Os gráficos com os *boxplots* representando os 10 RMSE obtidos em cada configuração estão representados nas Figuras 4.8, para uma camada oculta, e 4.9, para duas camadas ocultas. A melhor configuração obtida foi com apenas uma camada oculta e 12 neurônios. Porém, não há uma grande influência nos resultados com a variação de camadas ou neurônios.

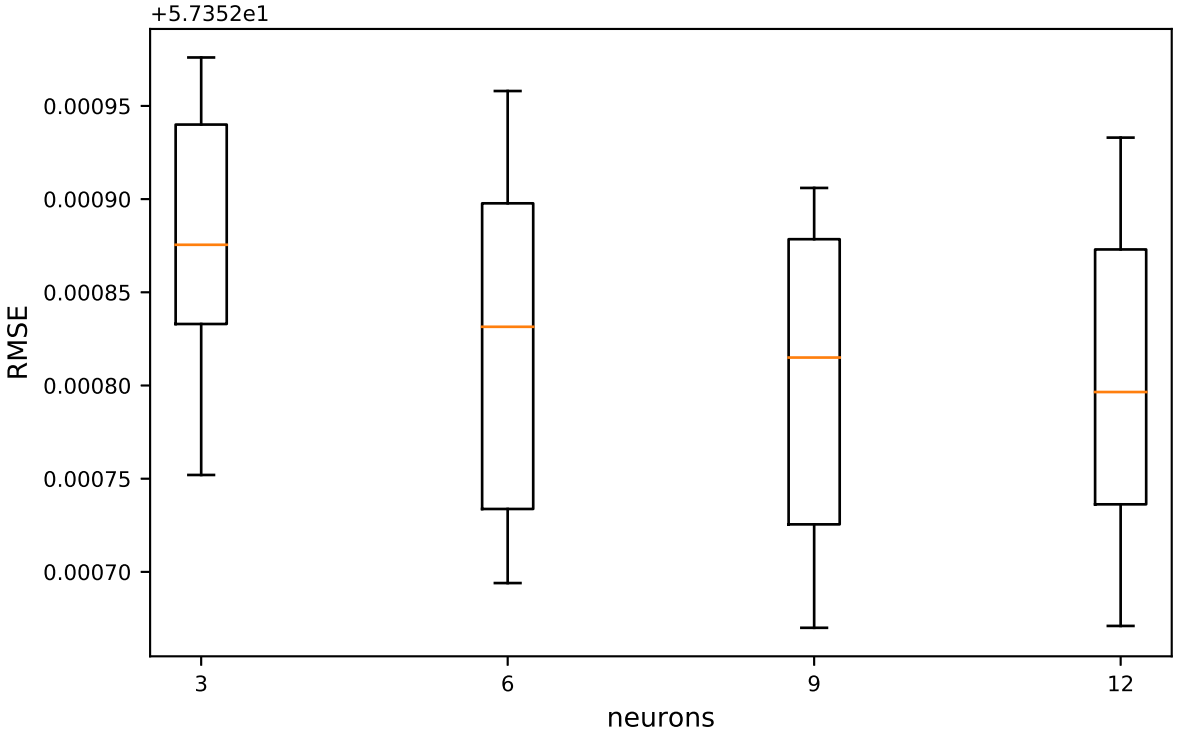

Figura 4.8: Rede Neural com uma camada oculta para *Dataset* II

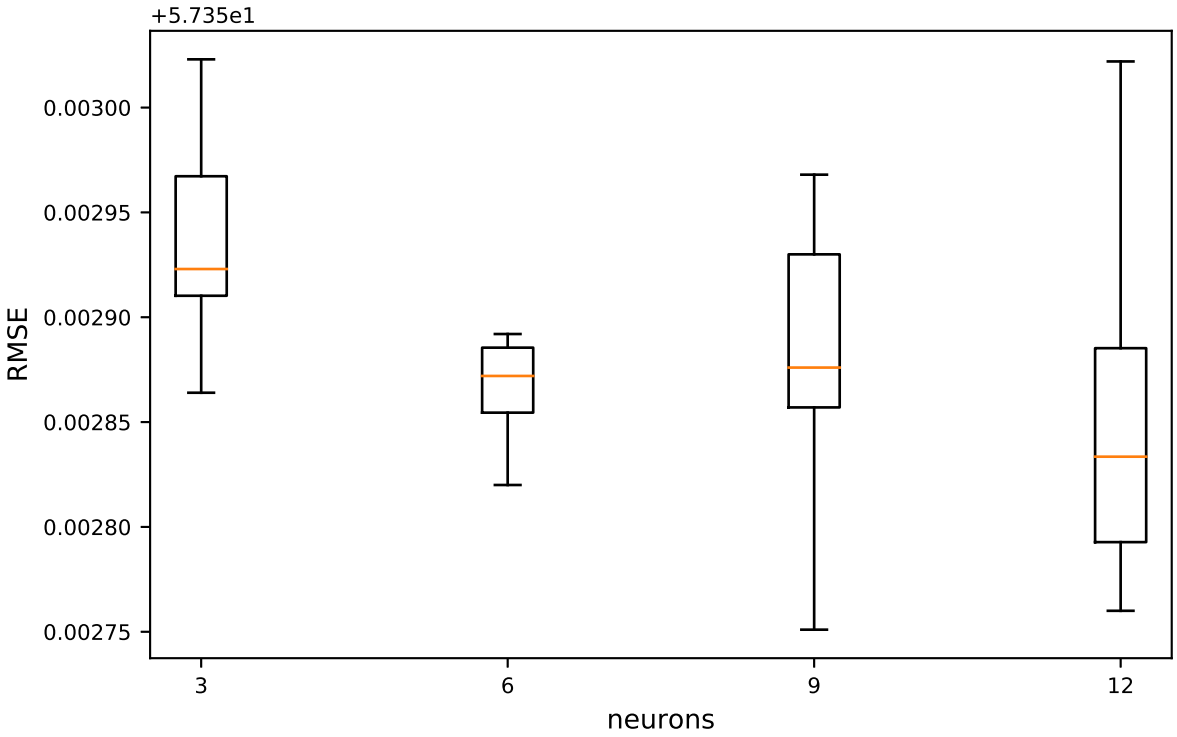

Figura 4.9: Rede Neural com duas camadas ocultas para *Dataset* II

Após encontrar a melhor configuração da estrutura da Rede Neural dentre as opções testadas, varia-se o fator de inércia k da MA entre 3, 8, 13, 18, 23 e 28. O gráfico contendo as médias dos 10 RMSE obtidos para cada k pode ser visualizado na Figura 4.10. O melhor fator de inércia k encontrado dentre as opções testadas foi com o valor de 3. É possível observar que, quanto menor o k, menores os erros obtidos.

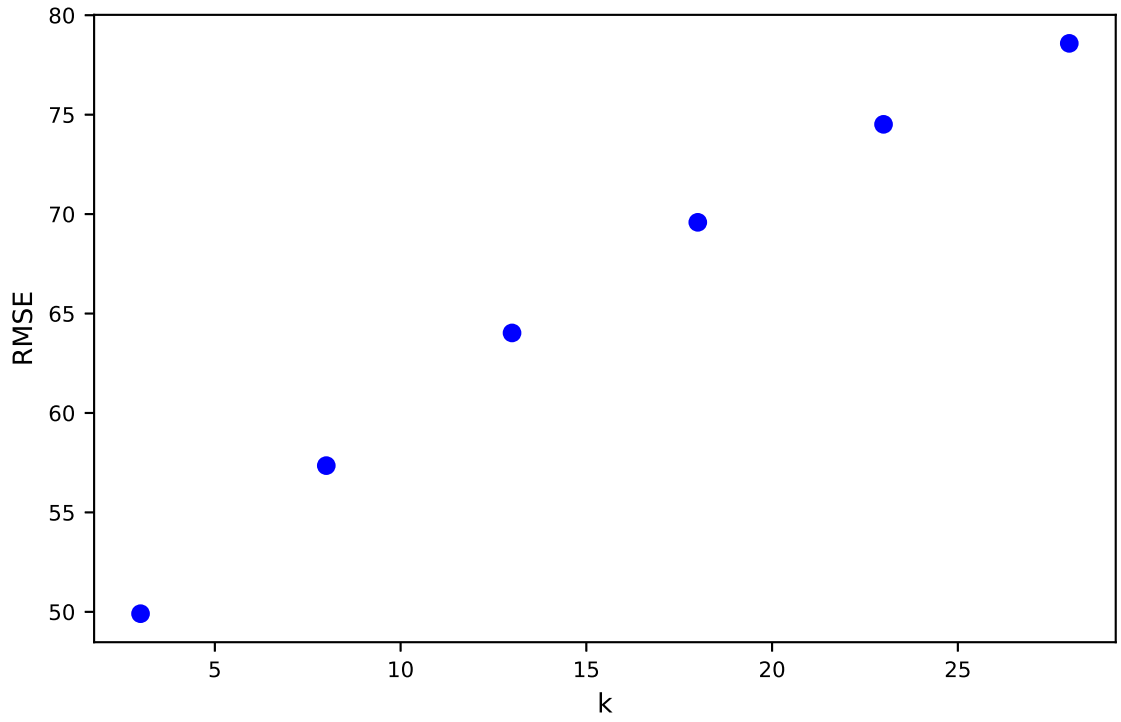

Figura 4.10: Desempenho da Rede Neural por fator de inércia da Média Móvel Exponencial para *Dataset* II

No seguinte passo, varia-se o tamanho w das janelas geradas entre 2, 7, 12, 17, 22 e 27. Esse valor representa o número de entradas ou de neurônios na camada de entrada da Rede Neural. O gráfico contendo as médias dos 10 RMSE obtidos para cada w pode ser visualizado na Figura 4.11. O melhor tamanho para janela w encontrado dentre as opções testadas foi com o valor de 2. É possível observar que, quanto menor o w, menores os erros obtidos.

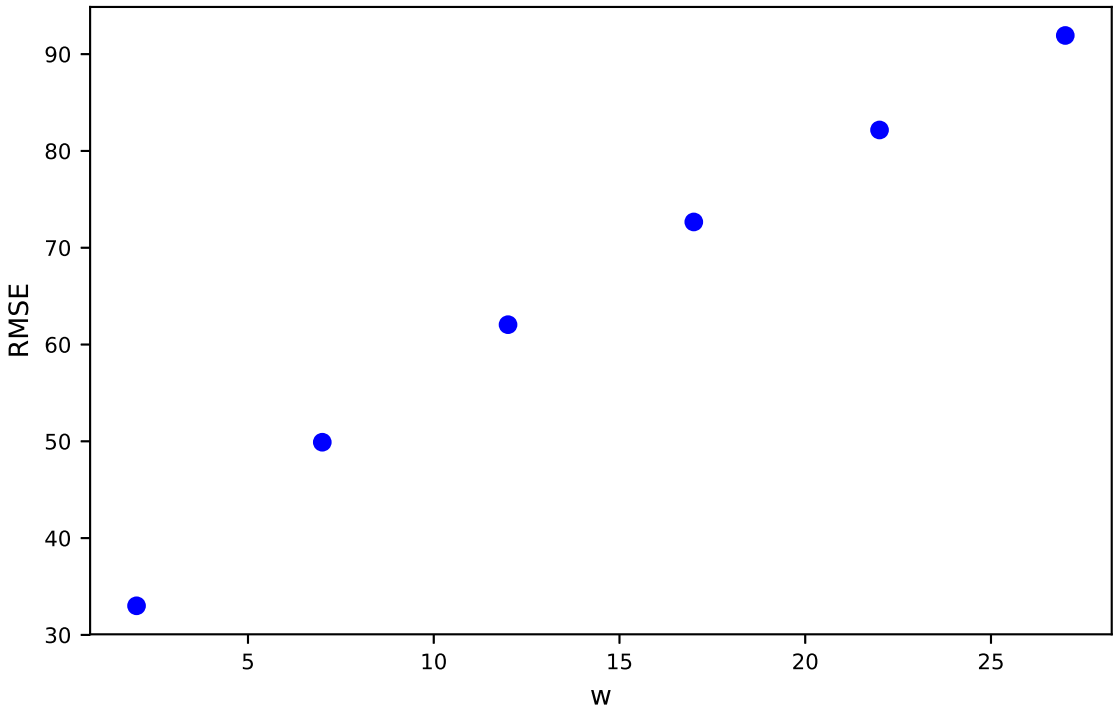

Figura 4.11: Desempenho da Rede Neural por tamanho de janela para *Dataset* II

Tendo sido encontrada a melhor configuração, ou seja, a que obteve o menor RMSE com a utilização da normalização AN original no pré-processamento, realizam-se experimentos com os mesmos parâmetros para cada normalização descrita nesse trabalho, visando comparar os resultados de forma justa. Os RMSE obtidos para cada normalização, assim como o obtido com a utilização do ARIMA, estão na Tabela 4.2. Lembrando que os RMSE representados na tabela são a média dos 10 RMSE obtidos ao rodar os experimentos com as 10 sementes fixas anteriormente, com exceção do ARIMA, cujo resultado é único por *dataset*.

Tabela 4.2: Desempenho dos algoritmos utilizados para previsões no *Dataset* II

Algoritmo	RMSE
ARIMA	30.71075
NN-MM	234.8197361
NN-DS	223.25956189999997
NN-ZS	404.1498575
NN-SW	31.7554871
NN-AN	33.0077409
NN-ANC	33.0077519
NN-ANS	**26.445085499999998**

O menor RMSE obtido foi de 26.4450855, com a ANS. Pode-se perceber que apenas esse método, ao ser aplicado junto à Rede Neural, superou o erro obtido com o *baseline* ARIMA. O NN-ANS (*Neural Network*-ANS) se mostrou 13,89% superior ao ARIMA, ou seja, obteve um RMSE 13,89% menor que o obtido com o *baseline*. Além disso, o método proposto ANS é superior em relação ao AN em 19,88% para esse *Dataset* com essas configurações.

Os outros ANs não conseguiram bater o SW com esse *Dataset*, diferente do que se pode observar nos exemplos mostrados no artigo (OGASAWARA *et al.*, 2010), em que o AN original sempre se mostrava superior tanto ao SW quanto ao ARIMA. Isso provavelmente se deve ao fato de que os períodos estão por minuto, assim como no Experimento I, enquanto no artigo são utilizados períodos maiores, como mensal ou trimestral, dessa forma a volatilidade é menor, o que talvez seja mais favorável à AN. Há também o fator de que são utilizados o número de neurônios como entrada igual ao tamanho da janela w, enquanto no artigo são fixados em três, independente do w.

Tanto a AN quanto a ANC, como previsto anteriormente, devido à pequena diferença entre suas normalizações (Tabela **??**), obtiveram erros muito semelhantes, estando praticamente idênticos nos gráficos. Portanto, mais uma vez, a solução ANC promete resolver o problema com médias nulas da AN original, afetando sensivelmente, de forma quase imperceptível, os resultados que se obteriam com a AN.

A remoção de *outliers* também é de grande importância nesse caso, já que elimina do conjunto de treino e de teste períodos de alta volatilidade, de difícil previsão, bastante comum em criptomoedas como o Bitcoin.

Observa-se que, como no artigo (OGASAWARA *et al.*, 2010), os algoritmos NN-ANs, NN-SW e ARIMA são os que obtém melhores resultados. Dentre os restantes, o método NN-DS se sai melhor, seguido de perto pelo NN-MM, provavelmente por que os dados de teste estão dentro do intervalo de mínimo e máximo obtidos do conjunto de treino, como pode ser visto na Figura 4.7, assim não ocorrendo perda de informações durante a normalização Min-Max. Como esperado, NN-ZS não consegue lidar bem com séries não-estacionárias, portanto seu RMSE se destaca negativamente.

Concluindo, verifica-se que o método proposto ANS é, mais uma vez, o melhor método para o *Dataset* II, superando tanto o *baseline* como o AN original. Porém, é importante testá-lo em mais *Datasets*, com períodos e aplicações diferentes, para validar a sua eficiência de forma generalizada, o que é feito nos experimentos seguintes. Até então, apenas dados financeiros e por minuto foram experimentados.

4.3.3 Experimento III

Neste experimento utiliza-se o *Dataset* III - contendo a quantidade de chuvas, medidas em milímetros, ocorridas por ano na cidade de Fortaleza, Ceará, entre 1850 e 1979, representado na Figura 4.12 - para treinamento e avaliação dos resultados. Compara-se os RMSE obtidos para cada uma das normalizações utilizadas como parte do pré-processamento no treinamento da rede, assim como o RMSE obtido com o ARIMA.

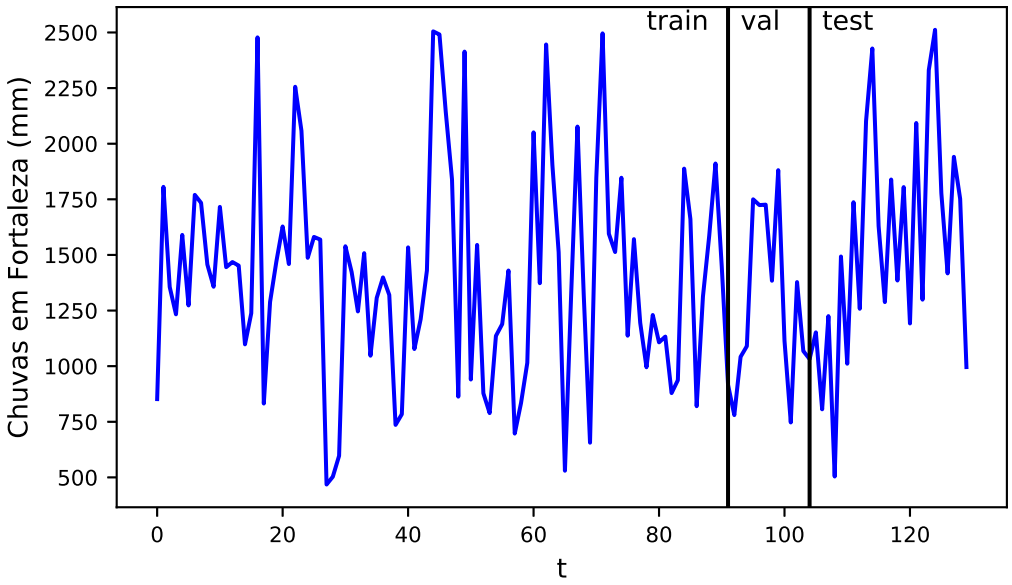

Figura 4.12: *Dataset* III, com cortes representando os conjuntos de treino, validação e teste

Assim como nos experimentos anteriores, foi realizada uma busca empírica para encontrar uma boa configuração de Rede Neural. Variou-se ela entre 1 e 2 camadas ocultas e entre 3, 6, 9 e 12 neurônios por camada oculta, utilizando a AN original como método de normalização durante o pré-processamento, com o $k = 7$ e o $w = 8$, e utilizando a Média Móvel Exponencial (EMA), como no experimento anterior. Os gráficos com os *boxplots* representando os 10 RMSE obtidos em cada configuração estão representados nas Figuras 4.13, para uma camada oculta, e 4.14, para duas camadas ocultas. A melhor configuração obtida foi com apenas uma camada oculta e 9 neurônios. Porém, não há uma grande influência nos resultados com a variação de camadas ou neurônios.

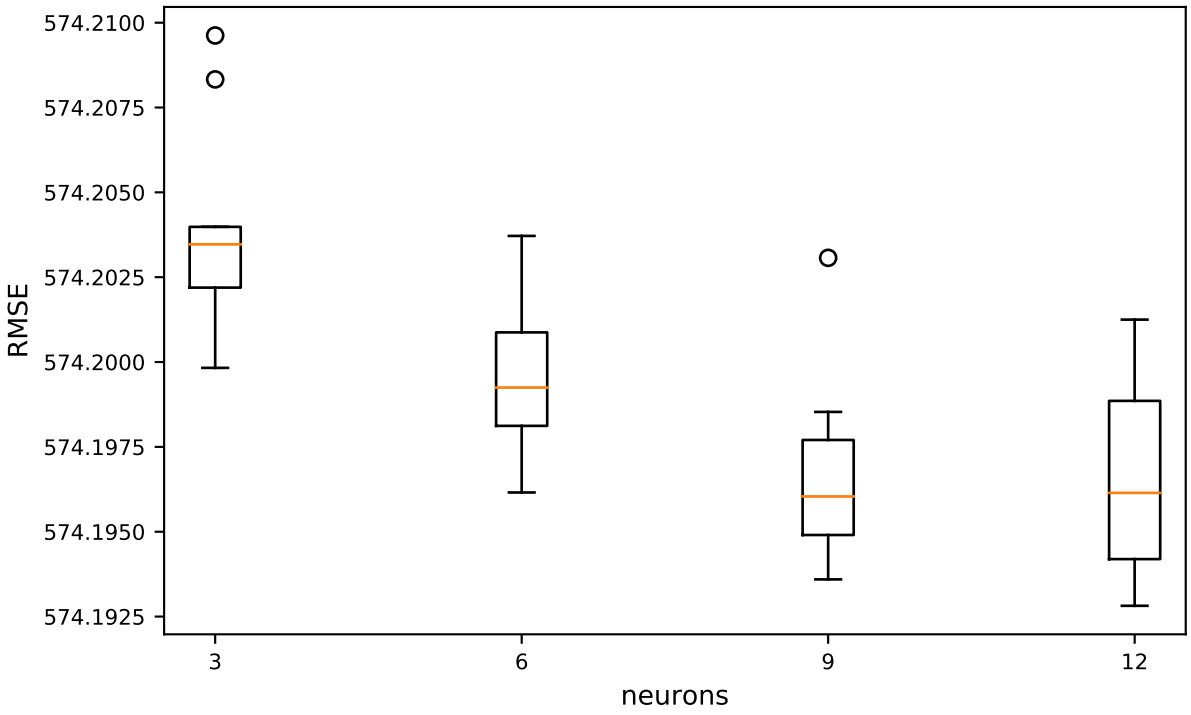

Figura 4.13: Rede Neural com uma camada oculta para *Dataset* III

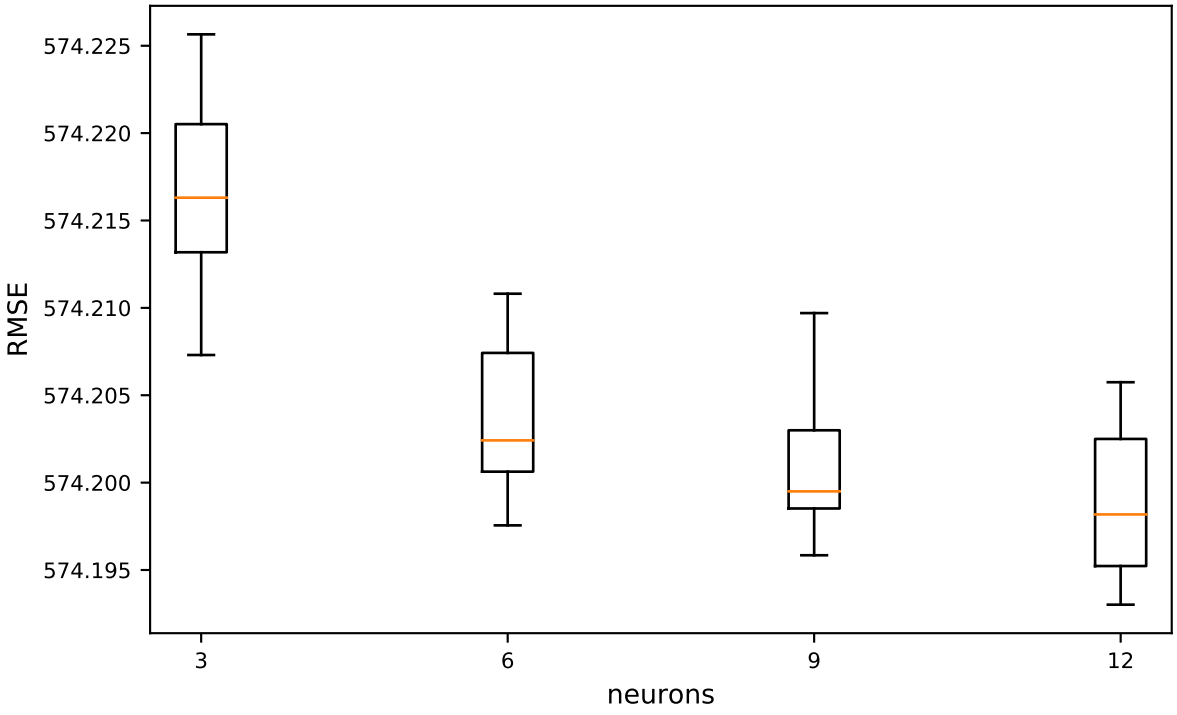

Figura 4.14: Rede Neural com duas camadas ocultas para *Dataset* III

Após encontrar a melhor configuração da estrutura da Rede Neural dentre as opções testadas, varia-se o fator de inércia k da MA entre 3, 8, 13, 18, 23 e 28. O gráfico contendo as médias dos 10 RMSE obtidos para cada k pode ser visualizado na Figura 4.15. O melhor fator de inércia k encontrado dentre as opções testadas foi com o valor de 23. Esse experimento se difere dos anteriores por haver uma diminuição do RMSE com um fator k bem maior. Isso ocorre, provavelmente por que o *dataset* possui uma volatilidade enorme, fazendo-se necessária uma média móvel que englobe uma quantidade de períodos maior, de modo a capturar uma inércia da série mais longa e mais estável ou menos variável.

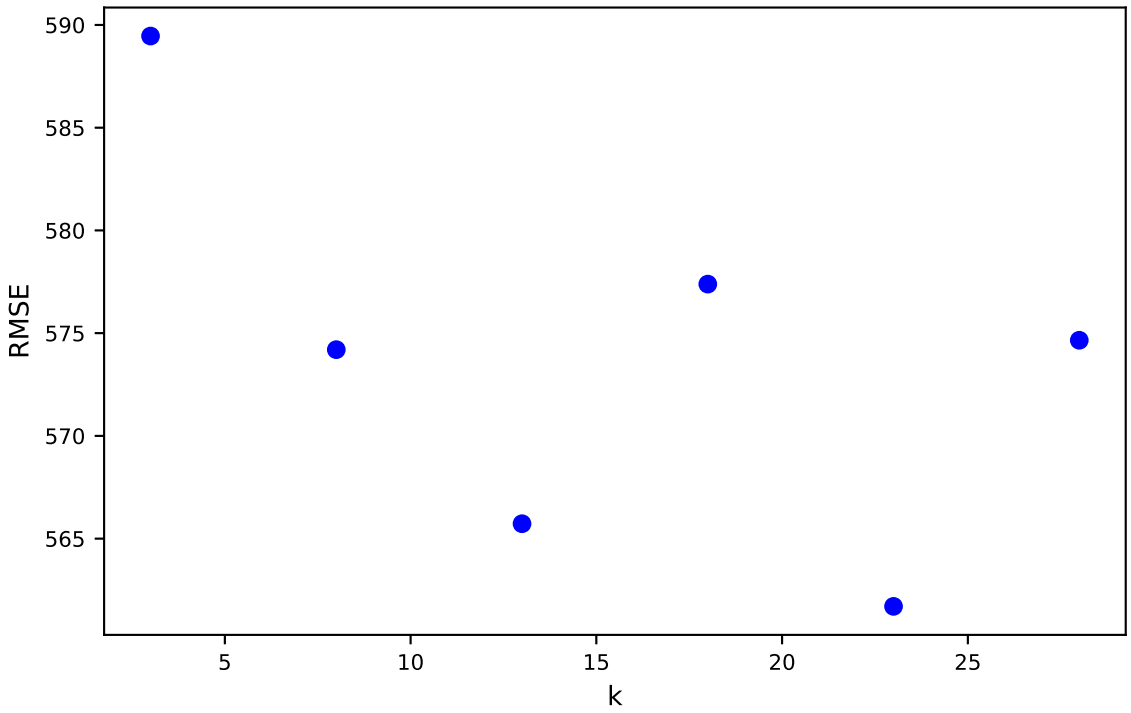

Figura 4.15: Desempenho da Rede Neural por fator de inércia da Média Móvel Exponencial para *Dataset* III

No seguinte passo, varia-se o tamanho w das janelas geradas entre 2, 7, 12, 17, 22 e 27. Esse valor representa o número de entradas ou de neurônios na camada de entrada da Rede Neural. O gráfico contendo as médias dos 10 RMSE obtidos para cada w pode ser visualizado na Figura 4.16. O melhor tamanho para janela w encontrado dentre as opções testadas foi com o valor de 2. É possível observar que, quanto menor o w, menores os erros obtidos.

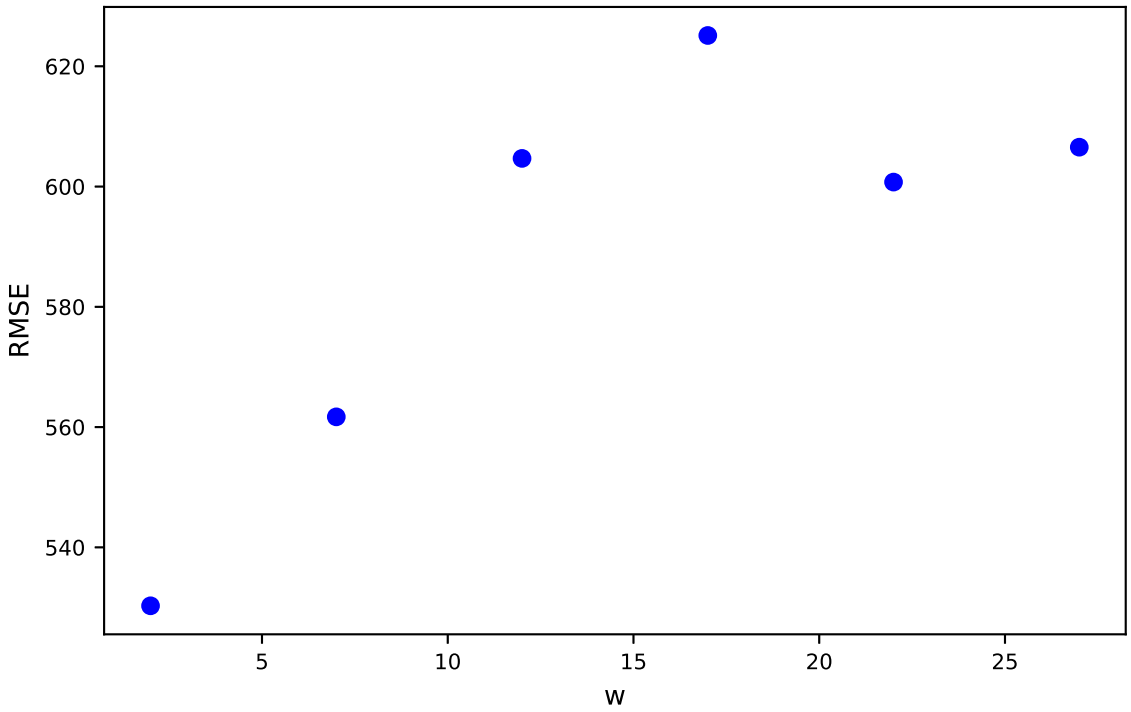

Figura 4.16: Desempenho da Rede Neural por tamanho de janela para *Dataset* III

Diferentemente dos experimentos anteriores, esse *Dataset* tem bem menos registros, assim como sua periodicidade é outra, por ano em vez de por minuto. Portanto, é de se esperar um comportamento diferente nos erros. Percebe-se, nas Figuras 4.15 e 4.16, uma tendência menos "comportada"ao se variar k e w. Isso se deve ao fato de existirem poucos dados para se treinar, assim como a dificuldade de se prever, já que os dados de chuva são por ano, e não por mês. Se fossem por mês, facilitaria a previsão, já que existem meses em que se chove mais que os outros e vice-versa. Enquanto uma medição por ano indica momentos de seca ou mais chuvas num prazo longo, o que é difícil de prever mesmo com a meteorologia atual.

Tendo sido encontrada a melhor configuração, ou seja, a que obteve o menor RMSE com a utilização da normalização AN original no pré-processamento, realizam-se experimentos com os mesmos parâmetros para cada normalização descrita nesse trabalho, visando comparar os resultados de forma justa. Os RMSE obtidos para cada normalização, assim como o obtido com a utilização do ARIMA, estão na Tabela 4.3. Lembrando que os RMSE representados na tabela são a média dos 10 RMSE obtidos ao rodar os experimentos com as 10 sementes fixas anteriormente, com exceção do ARIMA, cujo resultado é único por *dataset*.

Tabela 4.3: Desempenho dos algoritmos utilizados para previsões no *Dataset* III

Algoritmo	RMSE
ARIMA	**497.1257**
NN-MM	572.552784
NN-DS	587.0292371
NN-ZS	576.8336038
NN-SW	530.1379735
NN-AN	530.2831742
NN-ANC	530.2836633
NN-ANS	538.6819448

Através da Tabela 4.3, pode-se perceber que nenhum método obteve erro inferior ao obtido com o *baseline* ARIMA. O NN-AN se mostrou 6,67% inferior ao ARIMA, enquanto o ANS foi 8,36% pior, para essas configurações. Já o NN-SW superou todos os outros algoritmos, com exceção do ARIMA, porém ficou a frente do NN-AN e NN-ANC por uma diferença bem pequena.

Nesse experimento é possível observar uma diferença menor entre os resultados obtidos para todos os algoritmos. Isso provavelmente se deve ao fato de que esse *dataset* possui uma estacionariedade maior que os anteriores, o que permite que métodos tradicionais como ZS, DS e MM obtenham resultados melhores. Ou seja, essa série temporal tem sua média ao longo do tempo e variância mais comportadas, tanto no teste como no treino, como pode ser visto na Figura 4.12

O NN-DS se mostrou o pior de todos, porém, houve convergência da Rede Neural nesse experimento, trazendo resultados aceitáveis e próximos dos obtidos com os outros algoritmos. Isso provavelmente ocorre por que a normalização DS, ao ser aplicada nesse *Dataset* específico, gera valores com variações um pouco maiores que no experimento anterior, por exemplo, 2500 ficaria 0.25, enquanto um valor de 500 ficaria 0.05, como pode ser visto na Figura 4.12. Essa normalização acaba por diferenciar um pouco mais os valores de entrada, sendo suficiente para ocorrer a convergência da Rede Neural nessas configurações.

Em conclusão, verifica-se que o método ARIMA é o melhor método para o *Dataset* III. Observa-se também que os métodos NN-SW, NN-AN e NN-ANC se destacam entre os restantes e superam até o NN-ANS, o qual havia superado todos os outros nos experimentos anteriores. Com isso, pode-se criar a hipótese de que os métodos NN-AN, NN-ANC, NN-ANS e NN-SW se destacam quando a série temporal é não-estacionária ou possui um grau de estacionariedade bem baixo, como era o caso dos *Datasets* I e II. Já quando o grau de estacionariedade é alto, esses métodos acabam não tendo um desempenho satisfatório. Há também o fator de que são utilizados

o número de neurônios como entrada igual ao tamanho da janela w, enquanto no artigo são fixados em três, independente do w. Os *datasets* dos experimentos seguintes também possuem um grau de estacionariedade maior e servirão para validar ou não essa hipótese.

4.3.4 Experimento IV

Neste experimento utiliza-se o *Dataset* IV - contendo o volume de vazões médias mensais, medidas em metros cúbicos por segundo, ocorridas na usina hidrelétrica de FURNAS, entre janeiro de 1931 e dezembro de 1978, representado na Figura 4.17 - para treinamento e avaliação dos resultados. Compara-se os RMSE obtidos para cada uma das normalizações utilizadas como parte do pré-processamento no treinamento da rede, assim como o RMSE obtido com o ARIMA.

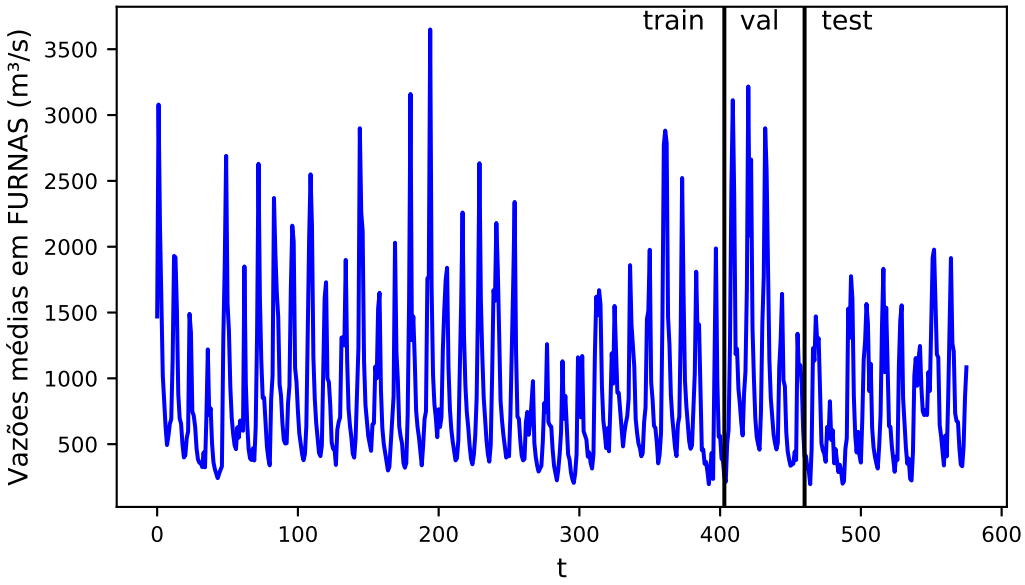

Figura 4.17: *Dataset* IV, com cortes representando os conjuntos de treino, validação e teste

Assim como nos experimentos anteriores, foi realizada uma busca empírica para encontrar uma boa configuração de Rede Neural. Variou-se ela entre 1 e 2 camadas ocultas e entre 3, 6, 9 e 12 neurônios por camada oculta, utilizando a AN original como método de normalização durante o pré-processamento, com o $k = 7$ e o $w = 8$, e utilizando a Média Móvel Exponencial (EMA), como no experimento anterior. Os gráficos com os *boxplots* representando os 10 RMSE obtidos em cada configuração estão representados nas Figuras 4.18, para uma camada oculta, e 4.19, para duas camadas ocultas. A melhor configuração obtida foi com apenas uma camada oculta

e 6 neurônios. Porém, não há uma grande influência nos resultados com a variação de camadas ou neurônios.

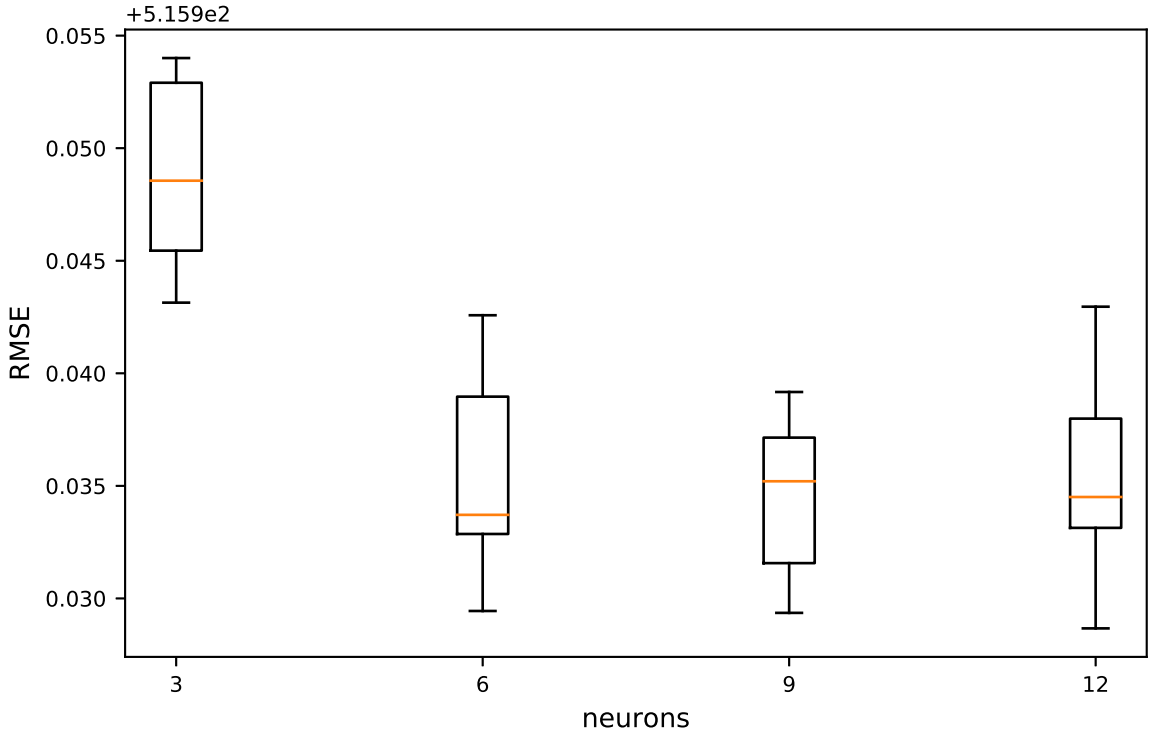

Figura 4.18: Rede Neural com uma camada oculta para *Dataset* IV

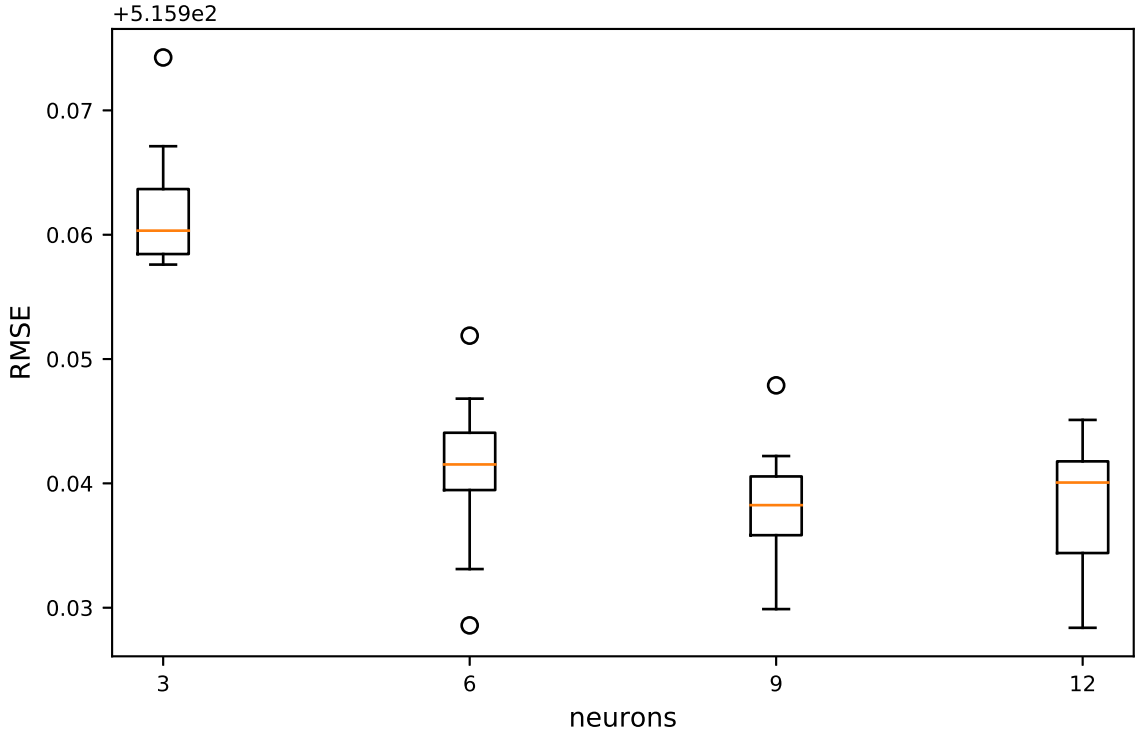

Figura 4.19: Rede Neural com duas camadas ocultas para *Dataset* IV

Após encontrar a melhor configuração da estrutura da Rede Neural dentre as opções testadas, varia-se o fator de inércia k da MA entre 3, 8, 13, 18, 23 e 28. O gráfico contendo as médias dos 10 RMSE obtidos para cada k pode ser visualizado na Figura 4.20. O melhor fator de inércia k encontrado dentre as opções testadas foi com o valor de 28. Esse experimento se difere dos dois primeiros e se iguala ao anterior por haver uma diminuição do RMSE com um aumento do fator k. Isso ocorre, provavelmente por que esse *dataset* também possui uma forte volatilidade, fazendo-se necessária uma média móvel que englobe uma quantidade de períodos maior, de modo a capturar uma inércia da série mais longa e mais estável ou menos variável.

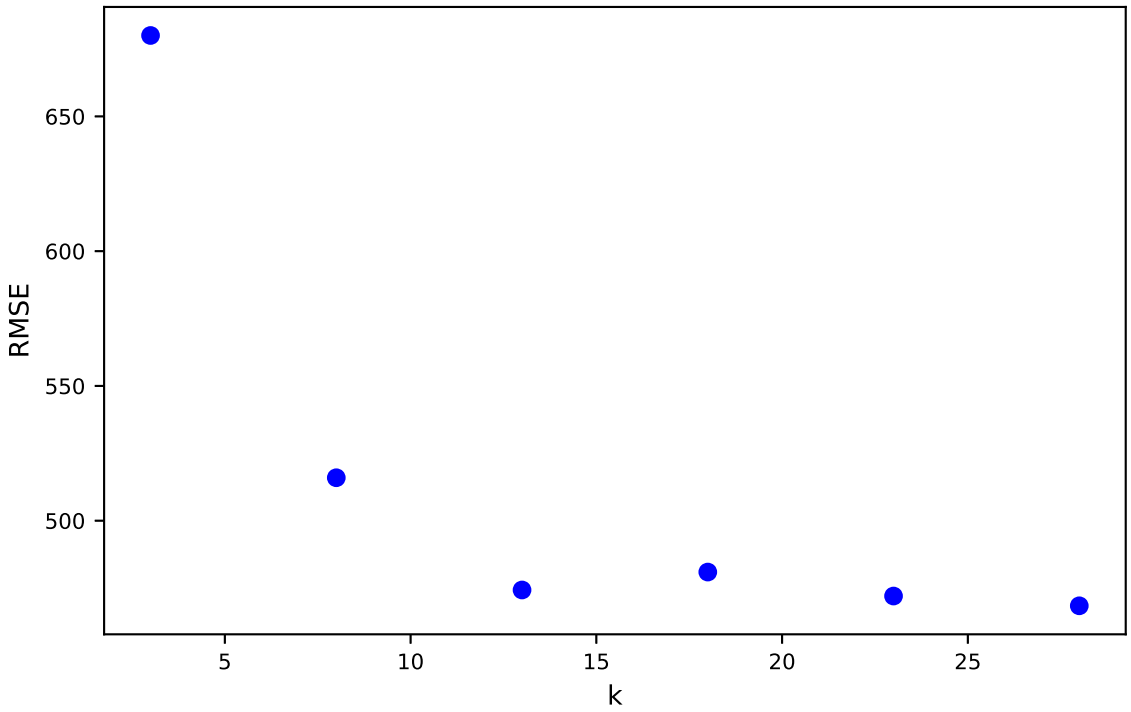

Figura 4.20: Desempenho da Rede Neural por fator de inércia da Média Móvel Exponencial para *Dataset* IV

No seguinte passo, varia-se o tamanho w das janelas geradas entre 2, 7, 12, 17, 22 e 27. Esse valor representa o número de entradas ou de neurônios na camada de entrada da Rede Neural. O gráfico contendo as médias dos 10 RMSE obtidos para cada w pode ser visualizado na Figura 4.21. O melhor tamanho para janela w encontrado dentre as opções testadas foi com o valor de 22. É possível observar uma certa aleatoriedade nos resultados obtidos ao se variar w, onde apenas o tamanho de janela $w = 22$ se destacou. Isso se deve a uma característica intrínseca deste *dataset*, onde apenas janelas com esse tamanho representam bem *features* necessárias para realizar previsões melhores.

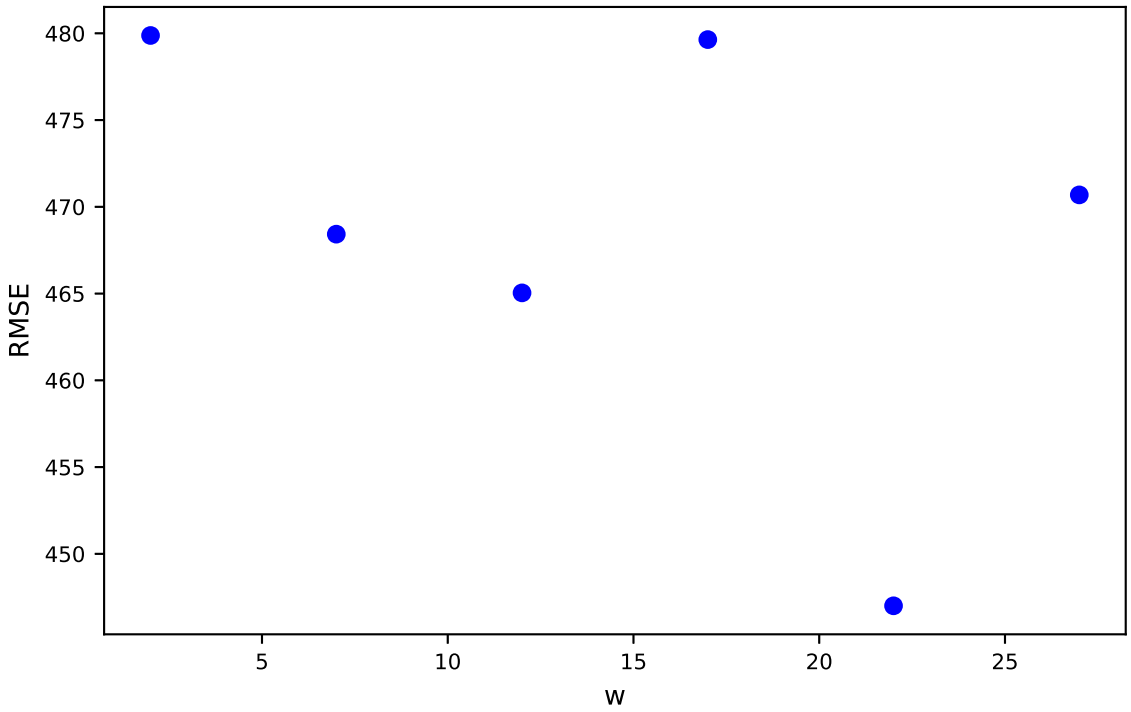

Figura 4.21: Desempenho da Rede Neural por tamanho de janela para *Dataset* IV

Assim como no experimento anterior, esse *Dataset* tem menos registros, sua periodicidade também é outra, por mês (no artigo do Ogasawara também há exemplo por mês) em vez de por minuto ou ano. Percebe-se também uma sazonalidade nos dados, já que eles são mensais e sabidamente há uma incidência maior de chuvas durante o verão no Brasil, as quais aumentam a vazão em FURNAS nesse período.

Tendo sido encontrada a melhor configuração, ou seja, a que obteve o menor RMSE com a utilização da normalização AN original no pré-processamento, realizam-se experimentos com os mesmos parâmetros para cada normalização descrita nesse trabalho, visando comparar os resultados de forma justa. Os RMSE obtidos para cada normalização, assim como o obtido com a utilização do ARIMA, estão na Tabela 4.4. Lembrando que os RMSE representados na tabela são a média dos 10 RMSE obtidos ao rodar os experimentos com as 10 sementes fixas anteriormente, com exceção do ARIMA, cujo resultado é único por *dataset*.

Tabela 4.4: Desempenho dos algoritmos utilizados para previsões no *Dataset* IV

Algoritmo	RMSE
ARIMA	**270.4243**
NN-MM	290.8909212
NN-DS	301.2198978
NN-ZS	315.08182289999996
NN-SW	274.2422956
NN-AN	447.00519629999997
NN-ANC	447.00448900000004
NN-ANS	275.617587

Através da Tabela 4.4, pode-se perceber que apenas o método ARIMA superou todos os outros, mais uma vez.Porém, a diferença foi menor dessa vez. O NN-ANS se mostrou 1,92% inferior ao ARIMA, enquanto o NN-AN foi 65,29% pior que o *baseline*, para essas configurações, sendo o pior algoritmo junto com o NN-ANC.

Nesse experimento é possível observar uma diferença menor entre os resultados obtidos para todos os algoritmos (devido a série temporal ter um grau de estacionariedade maior, assim como no experimento anterior), exceto para NN-AN e NN-ANC, que acabaram se saindo bem piores. Isso provavelmente se deve ao fato de que há sazonalidade nos dados, e isso atrapalha o desempenho da NN-AN e da NN-ANC. Há um exemplo no artigo em que existe a remoção de sazonalidade, justificada por assim revelar componentes não sazonais e ser mais fácil de realizar previsões na tendência e ciclos da série (OGASAWARA *et al.*, 2010). Há também o fator de que são utilizados o número de neurônios como entrada igual ao tamanho da janela w, enquanto no artigo são fixados em três, independente do w. Já o NN-SW superou todos os outros algoritmos, com exceção do ARIMA, porém ficou a frente do NN-ANS por uma diferença bem pequena.

Os métodos NN-MM e NN-DS se saíram melhores entre os restantes, já que ambos precisam do mesmo critério para obterem bons resultados: o máximo e mínimo global estar presente nos dados usados para treinamento da rede. Como se pode observar na Figura 4.17, esse critério é atendido. O método NN-ZS aparece logo em seguida, também por que essa série temporal tem sua média ao longo do tempo e variância mais comportadas, tanto no teste como no treino, possuindo então um grau de estacionariedade mais forte que nos *Datasets* I e II.

Em conclusão, verifica-se que o método ARIMA é o melhor método para o *Dataset* IV, superando todos os outros métodos. A hipótese de que os métodos NN-AN, NN-ANC, NN-ANS e NN-SW não se saem bem quando o grau de estacionariedade da série temporal é alto, citada no experimento anterior, é validada. Outra ob-

servação interessante nesse experimento é que o método ANS parece lidar bem com séries que possuem sazonalidade, enquanto o método original AN e o proposto ANC parecem se sair bem piores.

4.3.5 Experimento V

Neste experimento utiliza-se o *Dataset* V - contendo as variações percentuais dos fechamentos por minuto dos contratos futuros de MINI Dólar para Setembro de 2016, representado na Figura 4.22 - para treinamento e avaliação dos resultados. Compara-se os RMSE obtidos para cada uma das normalizações utilizadas como parte do pré-processamento no treinamento da rede, assim como o RMSE obtido com o ARIMA.

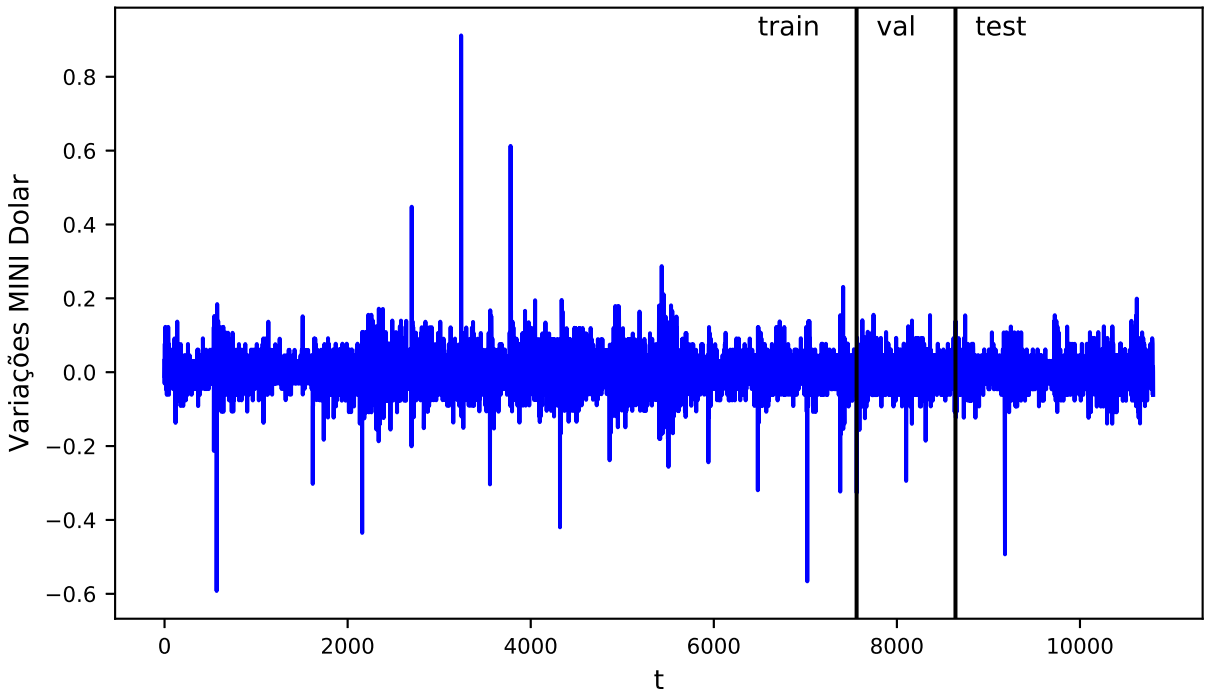

Figura 4.22: *Dataset* V, com cortes representando os conjuntos de treino, validação e teste

Novamente, afim de se encontrar uma boa configuração de Rede Neural, foi realizada uma busca empírica ao se variar entre 3, 6, 9 e 12 neurônios e entre 1 e 2 camadas ocultas, utilizando-se a AN original como método de normalização durante o pré-processamento, fixando o k em 7 e o w em 8, e utilizando a Média Móvel Exponencial, de acordo com o primeiro exemplo demonstrado no artigo (OGASAWARA *et al.*, 2010). Os gráficos com os *boxplots* representando os 10 RMSE obtidos em cada configuração estão representados nas Figuras 4.23, para uma camada oculta,

e 4.24, para duas camadas ocultas. A melhor configuração obtida foi com duas camadas ocultas e 3 neurônios. Neste experimento, nota-se uma grande influência nos resultados ao se variar o número de camadas ocultas, havendo uma diminuição de metade do erro ao se aumentar de uma para duas o número de camadas.

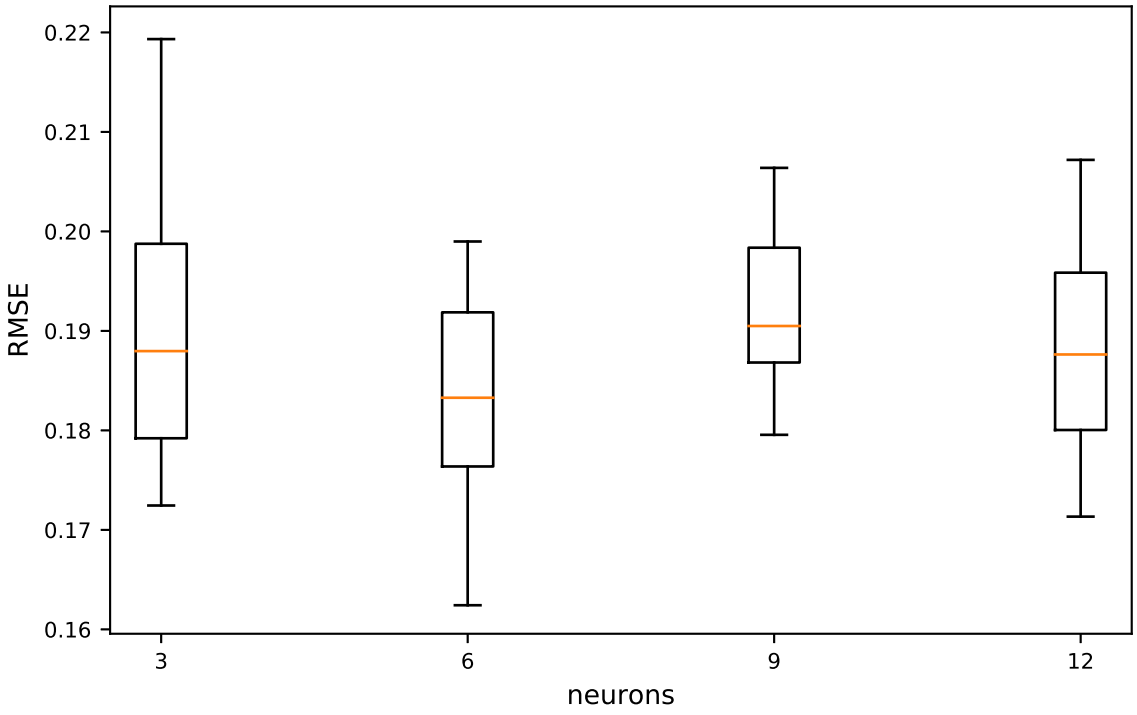

Figura 4.23: Rede Neural com uma camada oculta para *Dataset* V

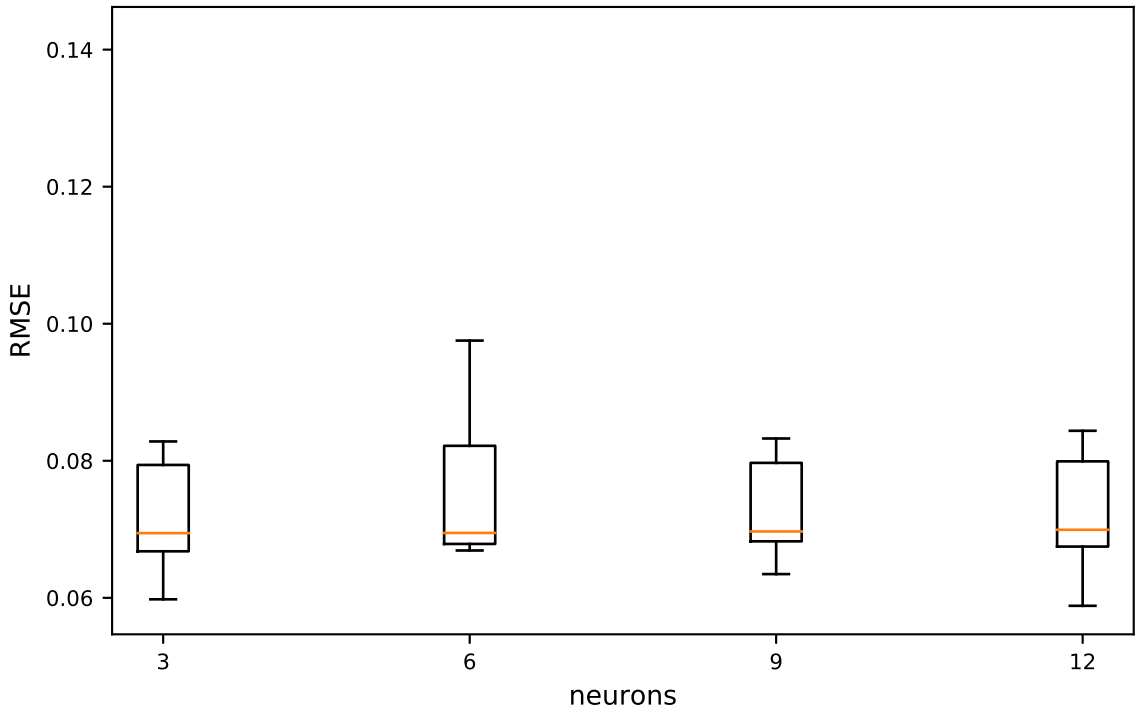

Figura 4.24: Rede Neural com duas camadas ocultas para *Dataset* V

Após encontrar a melhor configuração da estrutura da Rede Neural dentre as opções testadas, varia-se o fator de inércia k da MA entre 3, 8, 13, 18, 23 e 28. O gráfico contendo as médias dos 10 RMSE obtidos para cada k pode ser visualizado na Figura 4.25. O melhor fator de inércia k encontrado dentre as opções testadas foi com o valor de 3.

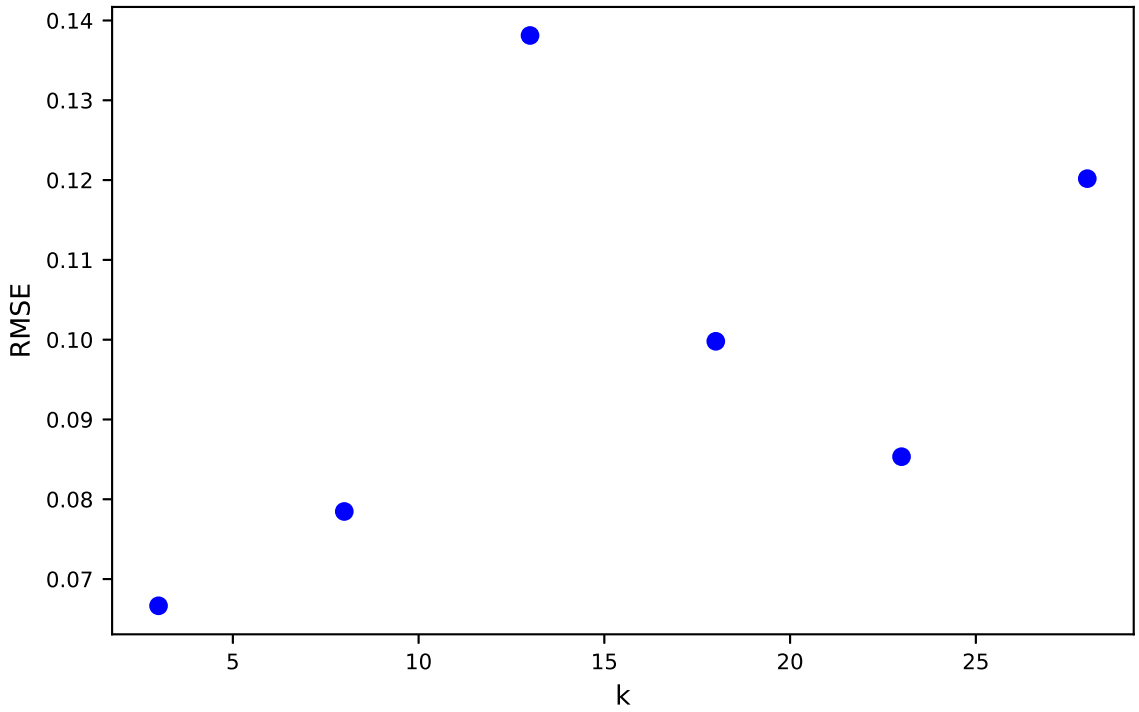

Figura 4.25: Desempenho da Rede Neural por fator de inércia da Média Móvel Exponencial para *Dataset* V

No seguinte passo, varia-se o tamanho w das janelas geradas entre 2, 7, 12, 17, 22 e 27. Esse valor representa o número de entradas ou de neurônios na camada de entrada da Rede Neural. O gráfico contendo as médias dos 10 RMSE obtidos para cada w pode ser visualizado na Figura 4.26. O melhor tamanho para janela w encontrado dentre as opções testadas foi com o valor de 12.

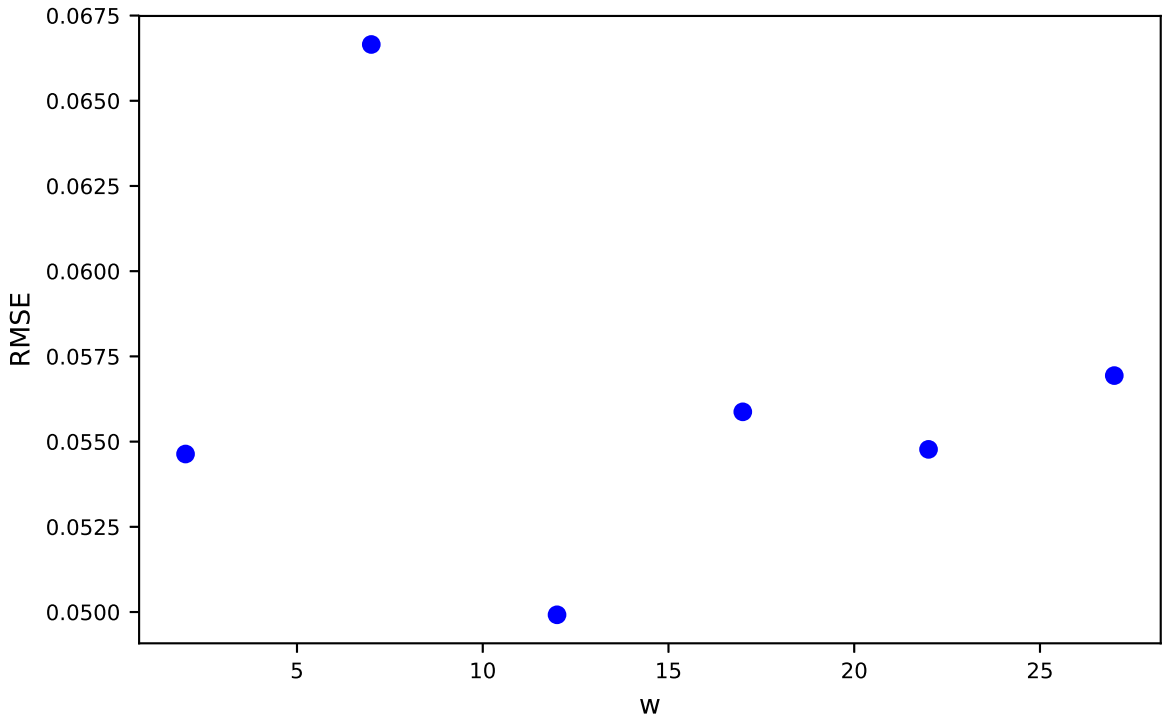

Figura 4.26: Desempenho da Rede Neural por tamanho de janela para *Dataset* V

Tendo sido encontrada a melhor configuração, ou seja, a que obteve o menor RMSE com a utilização da normalização AN original no pré-processamento, realizam-se experimentos com os mesmos parâmetros para cada normalização descrita nesse trabalho, visando comparar os resultados de forma justa. Os RMSE obtidos para cada normalização, assim como o obtido com a utilização do ARIMA, estão na Tabela 4.5. Lembrando que os RMSE representados na tabela são a média dos 10 RMSE obtidos ao rodar os experimentos com as 10 sementes fixas anteriormente, com exceção do ARIMA, cujo resultado é único por *dataset*.

Tabela 4.5: Desempenho dos algoritmos utilizados para previsões no *Dataset* V

Algoritmo	RMSE
ARIMA	0.0390725
NN-MM	0.032138
NN-DS	0.0321592
NN-ZS	**0.0321038**
NN-SW	0.032905
NN-AN	0.0504171
NN-ANC	1.0004692
NN-ANS	0.0348928

Através da Tabela 4.5, pode-se perceber que, diferente de todos os outros experimentos, o método NN-ZS se saiu melhor que os outros, sendo seguido de bem perto pelos NN-MM e NN-DS, todos superando o erro obtido com o *baseline* ARIMA. Os NN-SW e NN-ANS também superaram o ARIMA. O NN-ZS se mostrou 17,83% superior ao ARIMA, enquanto o NN-ANS foi apenas 10,69% superior ao *baseline*.

O motivo pelo qual os métodos mais tradicionais NN-MM, NN-ZS e NN-DS se saíram melhores se deve ao fato de que essa série temporal é a que possui o maior grau de estacionariedade dentre as experimentadas nesse capítulo, onde esses métodos conseguem os melhores resultados. Essa série, que originalmente é não-estacionária, representada pelo *Dataset* I, obtém uma estacionariedade elevada ao ser convertida para variações percentuais por minuto, já que a variância e média da nova série acaba por se manter quase uniforme.

A solução NN-SW é uma solução para séries não-estacionárias, não sendo destaque quando a série é estacionária ou próxima disso. Já os métodos NN-AN, NN-ANC e NN-ANS prometem resolver um problema da NN-SW, que é a não representação de volatilidade para cada janela, e também se destacam quando a série é não-estacionária, principalmente quando ela também possui heteroscedasticidade. O que não é o caso para essa série.

Os algoritmos NN-AN e NN-ANC são os únicos que obtiveram resultados piores que o ARIMA, provavelmente por que as médias móveis se aproximam de zero em diversos pontos. Já que essas normalizações realizam divisão, com as médias móveis como denominador, podem gerar janelas com valores muito grandes e outras bem pequenas, gerando uma faixa de valores bem extensa para serem normalizada entre -1 e 1, o que pode prejudicar as previsões. Há também o fator de que são utilizados o número de neurônios como entrada igual ao tamanho da janela w, enquanto no artigo são fixados em três, independente do w

A NN-ANC se saiu extremamente pior que todos os outros pois a constante definida para esse método foi o menor número inteiro possível: o valor 1. Como foi destacado no Capítulo 3, para *datasets* que possuam valores muito baixos, como é o caso do *Dataset* V, essa constante influencia fortemente nos cálculos da ANC, tornando-se quase desprezíveis os valores presentes na série temporal em relação a essa constante. Uma solução seria utilizar uma constante que possua uma ordem de grandeza menor que o elemento com menor ordem de grandeza presente na série temporal. Desse modo a influência da constante seria mínima e já resolveria o problema a que esse método se propõe, não permitir divisões por zero.

Ressalta-se também a importância da remoção de *outliers* para esse caso, já que esse *Dataset* pode sofrer variações bruscas de um dia para outro, pelos mesmos motivos destacados para o Experimento I.

Concluindo, verifica-se que o método NN-ZS é o melhor método para o *Data-*

set V, seguido de perto pelo NN-MM e NN-DS, superando tanto o *baseline* como o AN original e os dois métodos propostos. Mais uma vez, valida-se a hipótese de que os métodos propostos produzem os melhores resultados em séries temporais não-estacionárias e com heteroscedasticidade, comuns no mercado financeiro. Para séries temporais com um grau maior de estacionariedade eles possuem bons resultados, porém não se destacam. Isso comprova que o método AN pode ser utilizado, trazendo resultados relevantes, porém, para obter destaque depende da estacionariedade e heteroscedasticidade da série temporal analisada.

Capítulo 5

Conclusões

Neste capítulo são apontadas as conclusões obtidas com esse trabalho, discorrendo brevemente sobre o problema encontrado, as propostas apresentadas, resultados obtidos, contribuições e ideias de trabalhos futuros.

5.1 Considerações

Neste trabalho foi abordado o problema das médias nulas para a Normalização Adaptativa (AN) proposta por Eduardo Ogasawara (OGASAWARA *et al.*, 2010), utilizada em séries temporais. Sua proposta consiste em utilizar médias móveis e janelas deslizantes, para depois realizar uma normalização Min-Max global sobre todas as janelas deslizantes desassociadas geradas. Porém, quando a série temporal possuir médias nulas ao longo do tempo, a AN ficaria prejudicada, pois geraria valores indefinidos durante sua normalização. Duas propostas baseadas na AN foram sugeridas para solucionar essa questão.

Essas propostas foram comparadas em experimentos diversos com a AN original, assim como com outros tipos de normalização, ao serem assocadas à uma Rede Neural para realização de previsões em séries temporais.

5.2 Contribuições

Em vista do problema observado das médias móveis nulas para a Normalização Adaptativa, esse trabalho sugere duas novas propostas: Normalização Adaptativa Compensada (ANC) e Normalização Adaptativa por Subtração (ANS). A primeira sendo praticamente igual à AN original, apenas adicionando uma constante ao numerador e ao denominador, portanto, solucionando o problema das médias móveis nulas e trazendo resultados muito próximos dos obtidos com a AN original, como pode ser visto nos experimentos, em exceção do último cuja série possuía valores

com ordem de grandeza menor que a constante. Já a segunda altera a forma como é realizado o cálculo na AN original, utilizando subtração em vez de divisão, dessa forma os resultados ficam bem diferentes dos obtidos com a AN ou a ANC, sendo que, em apenas um dos experimentos, a ANS não superou a AN ou ANC.

Entretanto, tanto a AN original, como as ANS e ANC propostas exigem uma busca de parâmetros ideais: o tamanho de janela w e a ordem da média móvel k, as quais dependem de cada *Dataset* utilizado. Apesar de as outras normalizações associadas a uma NN também sofrerem certa influência devido ao tamanho da janela w, elas sofrem bem menos que as ANs, e não sofrem influência alguma da ordem k das médias móveis, simplesmente por que elas não se utilizam de médias móveis. A única influência da ordem k é no sentido de eliminar alguns poucos registros do *Dataset* se eles tiverem que ser calculados e isso, caso o *Dataset* seja pequeno, pode influenciar significativamente nos resultados. Se o *Dataset* já tiver todas as médias móveis originalmente, portanto, sem que haja a necessidade de cálculo dos mesmos, não haveria influência alguma nas outras normalizações além das ANs.

Como forma de contribuição também foram aplicados experimentos a *Datasets* com períodos menores, na ordem dos minutos, assim como períodos maiores, na ordem dos anos. Também foram analisadas os comportamentos das normalizações em diferentes graus de estacionariedade e heteroscedasticidade. Ou seja, esse trabalho apresentou uma soma ou continuação dos experimentos realizados por Ogasawara (OGASAWARA *et al.*, 2010).

5.3 Trabalhos futuros

Dentre possibilidades de pesquisas futuras, uma delas poderia ser utilizar mais *features* como entradas da Rede Neural, como valores de abertura, máximo, mínimo e volume para papéis do mercado financeiro (todas devidamente normalizadas com os diferentes métodos de normalização), além dos valores de fechamento que foram usados de forma exclusiva nesse trabalho para os *Datasets* financeiros (Mini contrato de dólar e Bitcoin).

Aplicações em outros *Datasets*, de diferentes domínios, assim como *Datasets* maiores, que exigiriam um poder computacional maior. Também seria interessante encontrar formas mais rápidas de achar os parâmetros ideais, além da tentativa e erro, que é uma forma muito custosa e que seria inviável para *Datasets* muito grandes.

Seria interessante também realizar uma simulação de compra e venda de papéis no mercado financeiro se utilizando das previsões realizadas com os métodos de normalização presentes nesse trabalho, como uma forma de comparação na eficiência dos métodos, além da já realizada por RMSE.

Os experimentos realizados nesse trabalho foram todos com as mesmas configurações de Rede Neural MLP, com apenas uma ou duas camadas intermediárias. Um trabalho futuro interessante seria aplicar esses métodos para Redes mais profundas (com maior número de camadas), assim como outros diferentes tipos de Rede Neural (LSTM, Convolucional, entre outras).

Referências Bibliográficas

MALKIEL, B. G., FAMA, E. F. "EFFICIENT CAPITAL MARKETS: A REVIEW OF THEORY AND EMPIRICAL WORK*", *The Journal of Finance*, v. 25, n. 2, pp. 383–417, maio 1970. ISSN: 00221082. doi: 10.1111/j.1540-6261.1970.tb00518.x. Disponível em: <http://doi.wiley.com/10.1111/j.1540-6261.1970.tb00518.x>.

FAMA, E. F. "Random Walks in Stock Market Prices", *Financial Analysts Journal*, v. 21, n. 5, pp. 55–59, set. 1965. ISSN: 0015-198X. doi: 10.2469/faj.v21.n5.55. Disponível em: <https://www.cfapubs.org/doi/abs/10.2469/faj.v21.n5.55>.

GIDÓFALVI, G. "Using news articles to predict stock price movements", 2001.

KHADJEH NASSIRTOUSSI, A., AGHABOZORGI, S., YING WAH, T., et al. "Text mining for market prediction: A systematic review", *Expert Systems with Applications*, v. 41, n. 16, pp. 7653–7670, nov. 2014. ISSN: 0957-4174. doi: 10.1016/j.eswa.2014.06.009. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0957417414003455>.

RACHLIN, G., LAST, M., ALBERG, D., et al. "ADMIRAL: A Data Mining Based Financial Trading System". In: *IEEE Symposium on Computational Intelligence and Data Mining, 2007. CIDM 2007*, pp. 720–725, mar. 2007. doi: 10.1109/CIDM.2007.368947.

ATSALAKIS, G. S., VALAVANIS, K. P. "Surveying stock market forecasting techniques – Part II: Soft computing methods", *Expert Systems with Applications*, v. 36, n. 3, Part 2, pp. 5932–5941, abr. 2009. ISSN: 0957-4174. doi: 10.1016/j.eswa.2008.07.006. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0957417408004417>.

ATSALAKIS, G., K, V. "Surveying stock market forecasting techniques - Part I: Conventional methods." pp. 49–104, jan. 2013.

PYLE, D. *Data preparation for data mining*, v. 1. morgan kaufmann, 1999.

TAN, P.-N., STEINBACH, M., KUMAR, V. *Introduction to Data Mining, (First Edition)*. Boston, MA, USA, Addison-Wesley Longman Publishing Co., Inc., 2005. ISBN: 0321321367.

MINING, W. I. D. "Data Mining: Concepts and Techniques", *Morgan Kaufinann*, 2006.

SHANKER, M., HU, M., HUNG, M. "Effect of data standardization on neural network training", *Omega*, v. 24, n. 4, pp. 385 – 397, 1996. ISSN: 0305-0483. doi: https://doi.org/10.1016/0305-0483(96)00010-2. Disponível em: <http://www.sciencedirect.com/science/article/pii/0305048396000102>.

NAYAK, S. C., MISRA, B. B., BEHERA, H. S. "Evaluation of normalization methods on neuro-genetic models for stock index forecasting". In: *2012 World Congress on Information and Communication Technologies*, pp. 602–607, Oct 2012. doi: 10.1109/WICT.2012.6409147.

OGASAWARA, E., MARTINEZ, L. C., OLIVEIRA, D. D., et al. "Adaptive Normalization: A novel data normalization approach for non-stationary time series". In: *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, jul. 2010. doi: 10.1109/IJCNN.2010.5596746.

TSAY, R. *Analysis of financial time series*. Wiley series in probability and statistics. 2. ed. ed. Hoboken, NJ, Wiley-Interscience, 2005. ISBN: 978-0-471-69074-0. Disponível em: <http://gso.gbv.de/DB=2.1/CMD?ACT=SRCHA&SRT=YOP&IKT=1016&TRM=ppn+483463442&sourceid=fbw_bibsonomy>.

GUJARATI, D. "Basic Econometrics". 2008.

KENDALL, M. G. M. G. *Time-series*. London : Charles Griffin, 1976. ISBN: 0852642415. Includes index.

CRYER, J., CHAN, K. *Time series analysis: with applications in R*. Springer, 2008.

NELSON, C., PLOSSER, C. "Trends and random walks in macroeconmic time series: Some evidence and implications", *Journal of Monetary Economics*, v. 10, n. 2, pp. 139–162, 1982. Disponível em: <https://EconPapers.repec.org/RePEc:eee:moneco:v:10:y:1982:i:2:p:139-162>.

PIERCE, D. A. "Relationships–and the Lack Thereof–Between Economic Time Series, with Special Reference to Money and Interest Rates", *Journal of*

the American Statistical Association, v. 72, n. 357, pp. 11–26, 1977. ISSN: 01621459. Disponível em: <http://www.jstor.org/stable/2286901>.

HAYKIN, S. S., HAYKIN, S. S., HAYKIN, S. S., et al. *Neural networks and learning machines*, v. 3. Pearson Education Upper Saddle River, 2009.

SALLES, R., ASSIS, L., GUEDES, G., et al. "A Framework for Benchmarking Machine Learning Methods Using Linear Models for Univariate Time Series Prediction". In: *The 2017 International Joint Conference on Neural Networks (IJCNN)*, 2017.

WOOLDRIDGE, J. *Introductory econometrics: A modern approach*. South Western Cengage Learning, 2009.

HASTIE, T., TIBSHIRANI, R., FRIEDMAN, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. Springer Series in Statistics. 2 ed. New York, Springer-Verlag, 2009. ISBN: 978-0-387-84857-0. Disponível em: <//www.springer.com/la/book/9780387848570>.

HORNIK, K., STINCHCOMBE, M., WHITE, H. "Multilayer feedforward networks are universal approximators", *Neural Networks*, v. 2, n. 5, pp. 359 – 366, 1989. ISSN: 0893-6080. doi: https://doi.org/10.1016/0893-6080(89)90020-8. Disponível em: <http://www.sciencedirect.com/science/article/pii/0893608089900208>.

LECUN, Y., BOTTOU, L., ORR, G. B., et al. "Efficient BackProp". In: *Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop*, pp. 9–50, London, UK, UK, 1998. Springer-Verlag. ISBN: 3-540-65311-2. Disponível em: <http://dl.acm.org/citation.cfm?id=645754.668382>.

BLUM, A., RIVEST, R. L. "Training a 3-node Neural Network is NP-complete". In: *Proceedings of the First Annual Workshop on Computational Learning Theory*, COLT '88, pp. 9–18, San Francisco, CA, USA, 1988. Morgan Kaufmann Publishers Inc. Disponível em: <http://dl.acm.org/citation.cfm?id=93025.93033>.

GLOROT, X., BENGIO, Y. "Understanding the difficulty of training deep feedforward neural networks". In: Teh, Y. W., Titterington, M. (Eds.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence*

and Statistics, v. 9, *Proceedings of Machine Learning Research*, pp. 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. Disponível em: <http://proceedings.mlr.press/v9/glorot10a.html>.

SHALABI, L. A., SHAABAN, Z. "Normalization as a Preprocessing Engine for Data Mining and the Approach of Preference Matrix". In: *2006 International Conference on Dependability of Computer Systems*, pp. 207–214, May 2006. doi: 10.1109/DEPCOS-RELCOMEX.2006.38.

SOLA, J., SEVILLA, J. "Importance of input data normalization for the application of neural networks to complex industrial problems", *IEEE Transactions on Nuclear Science*, v. 44, n. 3, pp. 1464–1468, Jun 1997. ISSN: 0018-9499. doi: 10.1109/23.589532.

LI, H.-F., LEE, S.-Y. "Mining Frequent Itemsets over Data Streams Using Efficient Window Sliding Techniques", *Expert Syst. Appl.*, v. 36, n. 2, pp. 1466–1477, mar. 2009. ISSN: 0957-4174. doi: 10.1016/j.eswa.2007.11.061. Disponível em: <http://dx.doi.org/10.1016/j.eswa.2007.11.061>.

HULL, J. *Options, Futures, and Other Derivatives,*. Pearson Education, 2010.

CHATFIELD, C. *The analysis of time series: an introduction*, v. 59. CRC press, 2004.

MOON, Y.-S., KIM, J. "Efficient Moving Average Transform-based Subsequence Matching Algorithms in Time-series Databases", *Inf. Sci.*, v. 177, n. 23, pp. 5415–5431, dez. 2007. ISSN: 0020-0255. doi: 10.1016/j.ins.2007.05.038. Disponível em: <http://dx.doi.org/10.1016/j.ins.2007.05.038>.

NEWTON, I. A. *"The" Principia: Mathematical Principles of Natural Philosophy*. University of California Pr, 1999.

GROSSBERG, S. "Nonlinear neural networks: Principles, mechanisms, and architectures", *Neural networks*, v. 1, n. 1, pp. 17–61, 1988.

MERMILLOD, M., BUGAISKA, A., BONIN, P. "The stability-plasticity dilemma: investigating the continuum from catastrophic forgetting to age-limited learning effects". In: *Front. Psychol.*, 2013.

JIANG, R., SZETO, K. Y. "Extraction of investment strategies based on moving averages: A genetic algorithm approach". In: *CIFEr*, 2003.

CHOY, K. "Outlier detection for stationary time series", v. 99, pp. 111–127, 12 2001.

TUKEY, J. "Exploratory data analysis", *Reading, MA*, 1977.

KVANLI, A. H., PAVUR, R. J., KEELING, K. B. *Concise managerial statistics*. Mason, Ohio : South-Western/Thomson Learning, 2006. ISBN: 0324223889 (InfoTrak college edition : package). "Supports Microsoft Excel, Minitab, and SPSS--Cover.

BOX, G. E., JENKINS, G. M., REINSEL, G. C. *Time series analysis: forecasting and control*. Wiley. com, 2013.

SHUMWAY, R. H., STOFFER, D. S. *Time series analysis and its applications: with R examples*. Springer, 2011.

YAO, J., TAN, C. L. "A case study on using neural networks to perform technical forecasting of forex", *Neurocomputing*, v. 34, n. 1, pp. 79–98, 2000.

HO, S., XIE, M., GOH, T. "A comparative study of neural network and Box-Jenkins ARIMA modeling in time series prediction", *Computers Industrial Engineering*, v. 42, n. 2, pp. 371–375, 2002.

GHIASSI, M., SAIDANE, H., ZIMBRA, D. "A dynamic artificial neural network model for forecasting time series events", *International Journal of Forecasting*, v. 21, pp. 341–362, 2005.

ZHANG, G., EDDY PATUWO, B., Y HU, M. "Forecasting with artificial neural networks:: The state of the art", *International journal of forecasting*, v. 14, n. 1, pp. 35–62, 1998.

KHASHEI, M., BIJARI, M. "A novel hybridization of artificial neural networks and ARIMA models for time series forecasting", *Applied Soft Computing*, v. 11, n. 2, pp. 2664–2675, 2011.

HYNDMAN, R., KHANDAKAR, Y. "Automatic Time Series Forecasting: The forecast Package for R", *Journal of Statistical Software, Articles*, v. 27, n. 3, pp. 1–22, 2008. ISSN: 1548-7660. doi: 10.18637/jss.v027.i03. Disponível em: <`https://www.jstatsoft.org/v027/i03`>.

HYNDMAN, R. J., ATHANASOPOULOS, G. *Forecasting: principles and practice*. OTexts, 2014.

HIPEL, K., MCLEOD, A. *Time Series Modelling of Water Resources and Environmental Systems*. Developments in Water Science. Elsevier Science, 1994. ISBN: 9780080870366. Disponível em: <`https://books.google.com.br/books?id=t1zG8OUbgdgC`>.

SRIVASTAVA, N., HINTON, G., KRIZHEVSKY, A., et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", *J. Mach. Learn. Res.*, v. 15, n. 1, pp. 1929–1958, jan. 2014. ISSN: 1532-4435. Disponível em: <http://dl.acm.org/citation.cfm?id=2627435.2670313>.

CARUANA, R., LAWRENCE, S., GILES, L. "Overfitting in Neural Nets: Backpropagation, Conjugate Gradient, and Early Stopping". In: *IN PROC. NEURAL INFORMATION PROCESSING SYSTEMS CONFERENCE*, pp. 402–408, 2000.

PRECHELT, L. "Automatic early stopping using cross validation: quantifying the criteria", *Neural Networks*, v. 11, n. 4, pp. 761–767, 1998.

NOWLAN, S. J., HINTON, G. E. "Simplifying Neural Networks by Soft Weight-sharing", *Neural Comput.*, v. 4, n. 4, pp. 473–493, jul. 1992. ISSN: 0899-7667. doi: 10.1162/neco.1992.4.4.473. Disponível em: <http://dx.doi.org/10.1162/neco.1992.4.4.473>.

LECUN, Y., KANTER, I., SOLLA, S. A. "Second Order Properties of Error Surfaces: Learning Time and Generalization". In: Lippmann, R. P., Moody, J. E., Touretzky, D. S. (Eds.), *Advances in Neural Information Processing Systems 3*, Morgan-Kaufmann, pp. 918–924, 1991. Disponível em: <http://papers.nips.cc/paper/314-second-order-properties-of-error-surfaces-learning-time-and-generaliza pdf>.

KINGMA, D. P., BA, J. "Adam: A Method for Stochastic Optimization", *CoRR*, v. abs/1412.6980, 2014.

RUDER, S. "An overview of gradient descent optimization algorithms", *CoRR*, v. abs/1609.04747, 2016. Disponível em: <http://arxiv.org/abs/1609.04747>.

ABADI, M., AGARWAL, A., BARHAM, P., et al. "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems". 2015. Disponível em: <http://download.tensorflow.org/paper/whitepaper2015.pdf>.

HUNTER, J. D. "Matplotlib: A 2D graphics environment", *Computing In Science & Engineering*, v. 9, n. 3, pp. 90–95, 2007. doi: 10.1109/MCSE.2007.55.