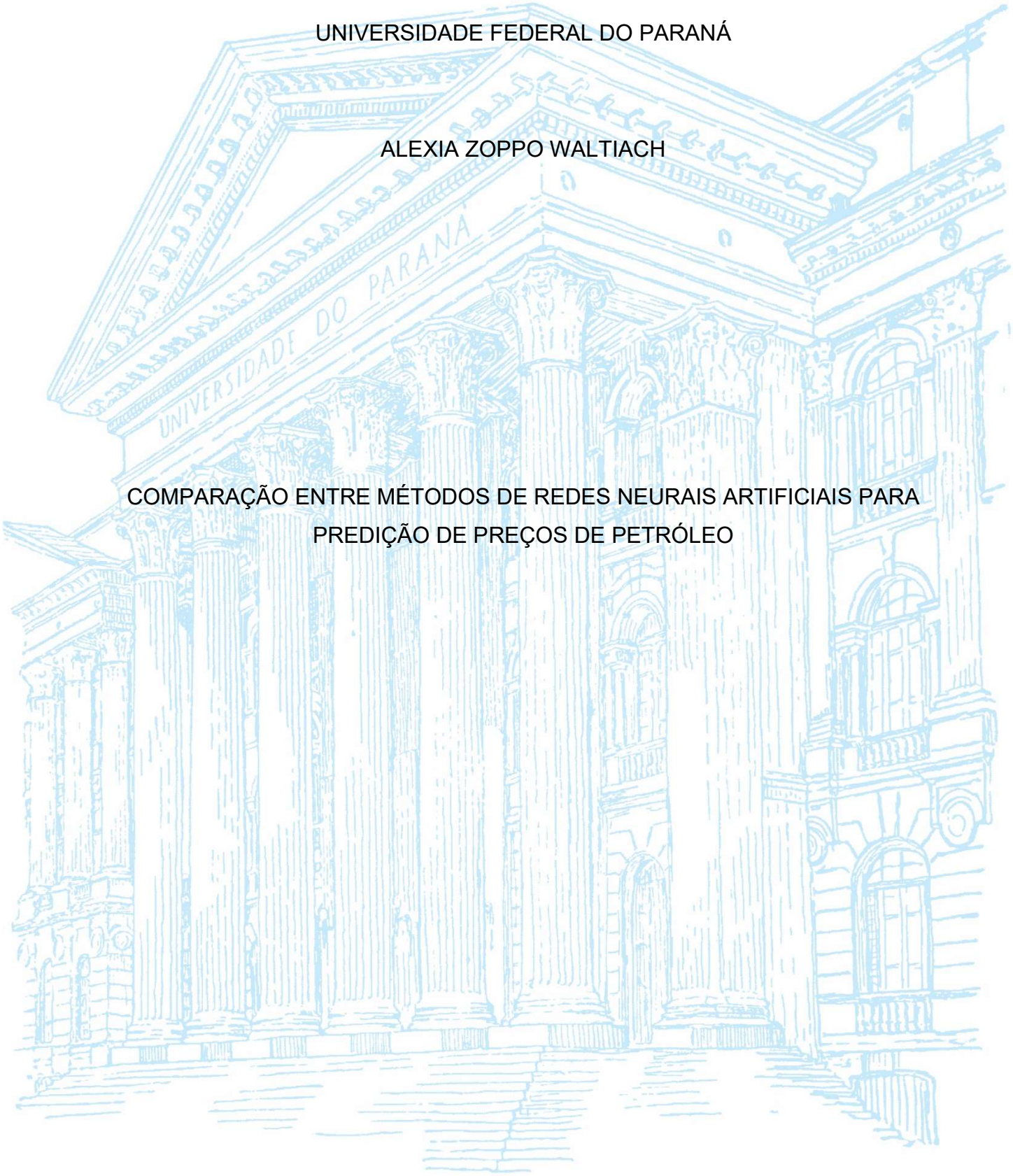
UNIVERSIDADE FEDERAL DO PARANÁ

ALEXIA ZOPPO WALTIACH



COMPARAÇÃO ENTRE MÉTODOS DE REDES NEURAIS ARTIFICIAIS PARA PREDIÇÃO DE PREÇOS DE PETRÓLEO

CURITIBA

2023

ALEXIA ZOPPO WALTIACH

COMPARAÇÃO ENTRE MÉTODOS DE REDES NEURAIS ARTIFICIAIS PARA PREDIÇÃO DE PREÇOS DE PETRÓLEO

TCC apresentada ao curso de Pós-Graduação em Engenharia de Produção, Setor de Tecnologia, Universidade Federal do Paraná, como requisito parcial à obtenção do título de Bacharel em Engenharia de Produção.

Orientadora: Dra .Mariana Kleina

CURITIBA

2023

DEDICATÓRIA

Dedico à Mariana Kleina, minha orientadora e guia nesta jornada acadêmica. Sua orientação, sabedoria e paciência foram fundamentais para a conclusão deste trabalho. À minha mãe, Beatriz, que sempre esteve ao meu lado, me apoiando e me inspirando com seu amor incondicional e sabedoria. Sua força e dedicação são um farol na minha vida, e este trabalho é dedicado a você, com profunda gratidão. A ambas, meu eterno agradecimento por serem as luzes que iluminaram o caminho até aqui.

AGRADECIMENTOS

Gostaria de expressar meus sinceros agradecimentos a todas as pessoas que contribuíram de alguma forma para a realização deste trabalho, pois sem o apoio delas, este TCC não teria sido possível.

Em primeiro lugar, minha mais profunda gratidão vai para minha orientadora, Mariana Kleina, cuja orientação, paciência e conhecimento foram fundamentais para o sucesso deste projeto. Seu comprometimento e dedicação em me guiar ao longo deste percurso acadêmico foram inestimáveis.

À minha mãe, Beatriz Zoppo, que é minha inspiração diária e a força motriz por trás de tudo o que faço. Seu amor e sabedoria infinita são um constante lembrete de que o esforço e a determinação podem superar qualquer obstáculo. Este trabalho é dedicado a você, com todo o meu amor e gratidão.

Também gostaria de agradecer a todos os meus professores e colegas que, de uma maneira ou de outra, contribuíram para o meu crescimento acadêmico e pessoal ao longo dos anos.

Por fim, agradeço a todos os amigos e familiares que me apoiaram, encorajaram e entenderam minha ausência durante essa jornada.

Da mesma forma que a autoridade divina foi respaldada por mitologias religiosas e a autoridade humana foi fundamentada pela narrativa liberal, a revolução tecnológica iminente pode estabelecer a preeminência dos algoritmos. (YUVAL NOAH HARARI, 2018)

RESUMO

O presente trabalho, de abordagem quantitativa, se propõe a investigar e desenvolver duas Redes Neurais Artificiais (RNAs) para a previsão de preços do petróleo, uma das *commodities* mais essenciais nos mercados globais. Por meio da revisão de literatura, optou-se pelo desenvolvimento das arquiteturas *Multi Layer Perceptron* (MLP) e *Long Short Term Memory* (LSTM). A primeira RNA adota uma estrutura de rede neural *feedforward*, enquanto a segunda utiliza uma rede neural recorrente para capturar padrões temporais complexos nos dados. A coleta e tratamento dos dados de preços do petróleo são discutidos em detalhes, juntamente com o processo de treinamento, teste e avaliação de desempenho das RNAs. Os resultados da previsão são comparados entre si para avaliação do melhor modelo para predição, levando em consideração o Erro Quadrático Médio. Os resultados demonstram a eficácia das RNAs na previsão de preços do petróleo, destacando as vantagens das RNAs recorrentes na captura de relações temporais. Este trabalho contribui para a compreensão do uso de RNAs na previsão de preços do petróleo e oferece percepções valiosas para investidores, empresas e órgãos reguladores que dependem de previsões precisas de preços de petróleo em seus processos de tomada de decisão.

Palavras-chave: Predição. Preço do Petróleo. Redes Neurais Artificiais. MLP. LSTM.

ABSTRACT

This qualitative study aims to investigate and develop two Artificial Neural Networks (ANNs) for predicting oil prices, one of the most crucial commodities in global markets. Through a literature review, the architectures chosen for development are the Multi Layer Perceptron (MLP) and the Long Short Term Memory (LSTM). The first ANN adopts a feedforward neural network structure, while the second employs a recurrent neural network to capture complex temporal patterns in the data. Data collection and preprocessing of oil price data are discussed in detail, along with the training, testing, and performance evaluation processes of the ANNs. The prediction results are compared to determine the best model for forecasting. The findings demonstrate the effectiveness of ANNs in oil price prediction, highlighting the advantages of recurrent ANNs in capturing temporal relationships. This work contributes to understanding the use of ANNs in oil price prediction and provides valuable insights for investors, businesses, and regulatory authorities reliant on accurate oil price forecasts for their decision-making processes.

Keywords: Forecasting. Crude Oil. Artificial Neural Network. MLP. LSTM.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS OU SIGLAS

RNA	- Rede Neural Artificial
MLP	- *Multi Layer Perceptron*
LSTM	- *Long Short Term Memory*

SUMÁRIO

1 INTRODUÇÃO

A volatilidade dos preços do petróleo não apenas impacta o setor de energia, mas também reverbera amplamente na economia global, afetando diretamente as indústrias que dependem de combustíveis fósseis, bem como a inflação e o poder de compra da população. Apesar da crescente importância de outras fontes energéticas – tais como as renováveis – o petróleo se manteve como principal fonte da matriz energética mundial, e só deverá perder o posto de fonte principal de energia quando houver restrição em sua oferta (BARROS, 2007).

A ascensão do petróleo como recurso essencial para o desenvolvimento econômico e industrial mundial foi impulsionada, principalmente, após a Segunda Guerra Mundial, quando a demanda por energia aumentou exponencialmente. Essa importância do petróleo se traduz em sua influência não apenas no crescimento econômico, mas também na geopolítica global. Conflitos internacionais frequentemente têm raízes na luta pelo controle das fontes energéticas, destacando ainda mais o papel estratégico do petróleo no cenário global (SOUZA, 2023).

O petróleo alimenta a sociedade industrial e o poderio econômico das grandes potências. A importância crescente da indústria do petróleo originou ainda inúmeros conflitos bélicos ao redor do mundo, evidenciando o papel essencial e estratégico da *commodity* para o desenvolvimento da economia moderna, sendo indissociável a ascensão de grandes potências e suas capacidades de controlar fontes energéticas. (SOUZA, 2023)

A indústria petrolífera internacional está sujeita a flutuações cíclicas dos preços do petróleo, que muitas vezes desencadeiam reestruturações significativas no setor (PEDROSA, CORRÊA, 2016). Portanto, é crucial o desenvolvimento de técnicas de previsão para antecipar essas oscilações e permitir uma tomada de decisão mais informada.

As RNAs são capazes de aprender a partir de dados históricos, e podem detectar padrões e tendências que podem passar despercebidos pelos métodos tradicionais de análises. Fato que comprova que o uso de Redes Neurais Artificiais se faz de tamanha importância na atuação como ferramenta de predição, podendo ser usada para pautar decisões estratégicas de negócios, diminuindo o risco de erros, corrigindo e evitando prejuízos. (AMBRÓSIO, 2002)

A tomada de decisão é um momento crucial no cotidiano de qualquer organização - o alto grau de complexidade e velocidade do fluxo decisório demandam

ações rápidas e precisas e é a qualidade dessas ações que embasarão as decisões a serem tomadas. Nesse contexto, tem-se como ferramenta de apoio técnicas de previsão, que objetivam reduzir risco nos momentos das tomadas de decisão, estabelecendo um melhor desempenho estratégico. Para isso, o uso de Redes Neurais Artificiais tem se concretizado como uma importante alternativa - podendo inferir relações não lineares entre as observações - em detrimento de técnicas estatísticas tradicionais.

Uma decisão assertiva coopera para a manutenção da competitividade da organização e é nesse espectro que se desenvolvem sistemas de predição - que se materializam como um norteador dos rumos dos negócios, sendo base do planejamento estratégico. Entretanto, é importante ressaltar que, mesmo tendo fatídica importância nos processos, a predição carrega erros - que devem ser considerados na coleta dos dados, no momento da escolha da metodologia de previsão e na tomada de decisão (MARTINS; LAUGENI, 2005).

De acordo com Wright (1985), os métodos de previsão de séries temporais se baseiam na suposição de que as observações passadas contêm todas as informações sobre o padrão de comportamento, entretanto com a mais sofisticada técnica de previsão é difícil utilizar somente dados históricos para prever tendências futuras e sazonalidades, sendo importante a análise multivariada considerando mais de um fator para mudanças da predição. Assim, o presente trabalho visa não só predizer o preço do petróleo como também entender fatores que podem influenciar a mudança desse preço.

Os preços do petróleo bruto têm uma forte correlação com a economia, podendo ser um importante indicador de desempenho econômico. A previsão do preço dessa *commodity* tem um escopo muito abrangente, podendo beneficiar várias empresas (do ramo petroleiro ou não), pessoas e governo, sendo um importante objeto de estudo e previsão.

1.1 JUSTIFICATIVA

Apesar de importantes crises do petróleo evidenciarem ao mundo as consequências de se ter uma economia pautada energeticamente por uma fonte esgotável e vulnerável a fortes oscilações no preço, esse hidrocarboneto se mantém como a fonte energética mais consumida do mundo. O presente trabalho se justifica,

então, em virtude da importância do preço do petróleo para movimentação da economia nacional e internacional, sendo imprescindível o estudo de sua variação e a utilização de técnicas de predição – para que se possa tomar ações estratégicas baseadas em preços futuros da *commodity*.

Cavalheiro *et al*. (2010) afirmam que o uso de Redes Neurais Artificiais em modelos de previsão tem se mostrado cada vez mais presente na literatura e nas modelagens empíricas. Ao utilizar RNAs, é possível realizar previsões mais precisas e com maior rapidez, o que é fundamental para os tomadores de decisão do mercado de petróleo. Além disso, as RNAs podem ser facilmente adaptadas para incorporar novos dados e informações, importante fator em análises multivariadas, tornando-as uma ferramenta flexível e atualizada

Com isso, um trabalho que explore a predição de preços do petróleo com o uso de RNAs é altamente pertinente e pode contribuir para aprimorar as estratégias de investimento e gestão de risco no mercado de *commodities*, beneficiando tanto os investidores como as empresas do setor.

Nesse sentido, o presente estudo busca oferecer uma contribuição valiosa ao fornecer *insights* sobre a aplicação de RNAs na previsão de preços do petróleo, possibilitando uma melhor compreensão das dinâmicas do mercado e ajudando a aprimorar a tomada de decisão em um contexto globalmente interconectado e complexo.

1.2 OBJETIVOS

1.2.1 Objetivo geral

Aplicar e comparar o desempenho de duas Redes Neurais Artificiais para predição de preço do petróleo.

1.2.2 Objetivos específicos

Para que o objetivo geral seja alcançado, faz-se necessário a conclusão dos seguintes objetivos específicos:

a) - Mapear na literatura o uso de Redes Neurais Artificiais, a fim de entender os tipos de redes mais utilizadas na predição de preço de petróleo;

b) - Definir dois tipos de Redes Neurais Artificiais para aplicação em um estudo de caso;
c) - Escolher uma base de dados com o preço do petróleo;
d) - Escolher parâmetros adequados para as Redes Neurais Artificiais utilizadas;
e) - Comparar o desempenho das duas redes por meio de uma medida de erro.

2 REVISÃO DE LITERATURA

O petróleo representou mais de 25% da matriz energética mundial em 2020 (IEA, 2020) e cerca de 35% da matriz energética brasileira em 2021 (BEN, 2022), se concretizando como a principal fonte de energia mundial e nacional. Além do viés energético, o petróleo ainda é matéria prima de diversos produtos como plástico e cosméticos e é nesse panorama que fica claro a importância de se criar ferramentas de previsão de preço de petróleo – visto que seu valor impacta na precificação de inúmeros outros produtos. Entretanto, por ser muito volátil e ser impactado por diversas variáveis – tais como demanda e situação geopolítica – o valor do petróleo é difícil de ser previsto.

Nesse contexto, de forma a justificar a presente pesquisa e buscar encontrar o que já foi produzido e estudado sobre o uso de Redes Neurais Artificiais na predição de preços de petróleo, realizou-se o estudo de revisão – que busca não só encontrar lacunas nas pesquisas como também justificar a pertinência do presente estudo. Os descritores usados são apresentados na Tabela 1, ressalta-se ainda que se filtrou somente pesquisas realizadas entre os anos de 2019 e junho de 2023 e os descritores precisaram estar no título, resumo ou palavras-chave.

Tabela 1 - Mapeamento dos trabalhos

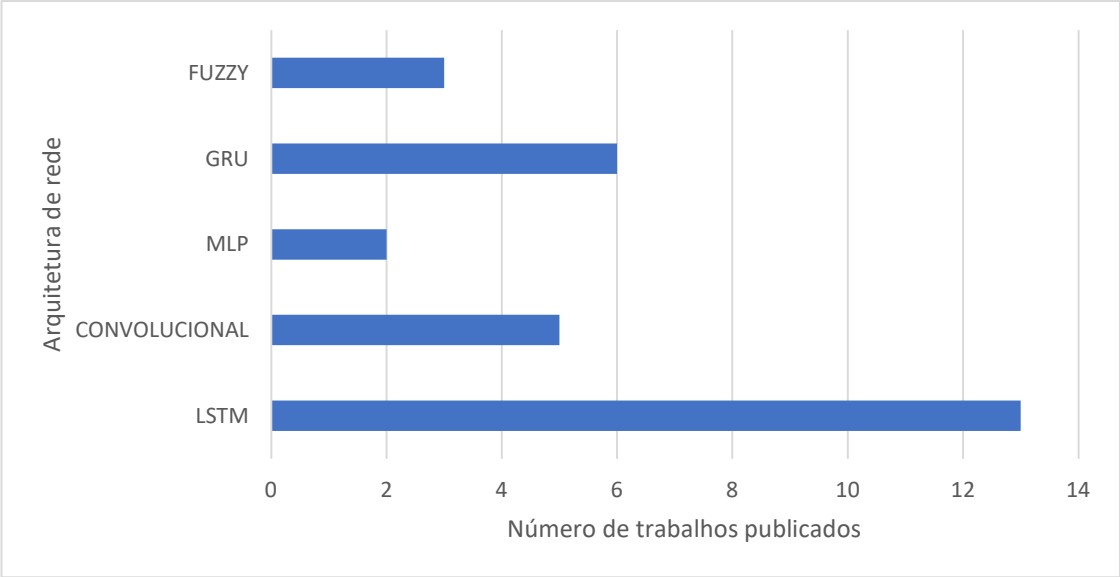
Bases	Trabalho	Descritores	Total
Science Direct	*Review Articles e Research Articles*	*"Artfificial Neural Network" AND ("crude oil" OR "Petroleum") AND "price prediction"*	14
SCOPUS	Todos	*"Artificial Neural Network" AND ("crude oil" OR "Petroleum") AND "price prediction"*	39
Total de documentos encontrados			53
Documentos em duplicidade e temas não pertinentes (-)			21
Total para leitura			32

Fonte: a autora (2022)

As Redes Neurais Artificiais têm sido utilizadas em diversas aplicações para modelar e prever problemas relacionados à predição de séries temporais nas mais distintas áreas do conhecimento (SIAMI-NAMINI *et al*., 2019). A principal razão para sua ampla aplicação reside na capacidade de generalização e processamento

temporal – fato que possibilita a resolução de problemas de distintas complexidades (FIORIN *et al*., 2011). Uma das dificuldades encontradas no uso de Redes Neurais Artificias é a escolha da arquitetura que melhor solucione a questão problema, por isso a necessidade de se aplicar diversos métodos de aprendizado e diferentes configurações de rede a fim de resolver um problema específico (MIRANDA, 2009). Nesse contexto, percebeu-se uma tendência a utilização de 5 principais tipos de Redes Neurais Artificiais, sendo eles: Rede Neural Convolucional, *Gated Recurrent Unit* (GRU), Neural *Fuzzy*, *Multi Layer Perceptron* (MLP) e *Long Short Term Memory* (LSTM). A Figura 1 representa o número de trabalhos publicados por cada tema encontrados na revisão de literatura de acordo com o tipo de RNA empregada.

Figura 1 - Trabalhos publicados sobre Redes Neurais Artificiais



Fonte: a autora (2023)

2.1 REDES NEURAIS ARTIFICIAIS

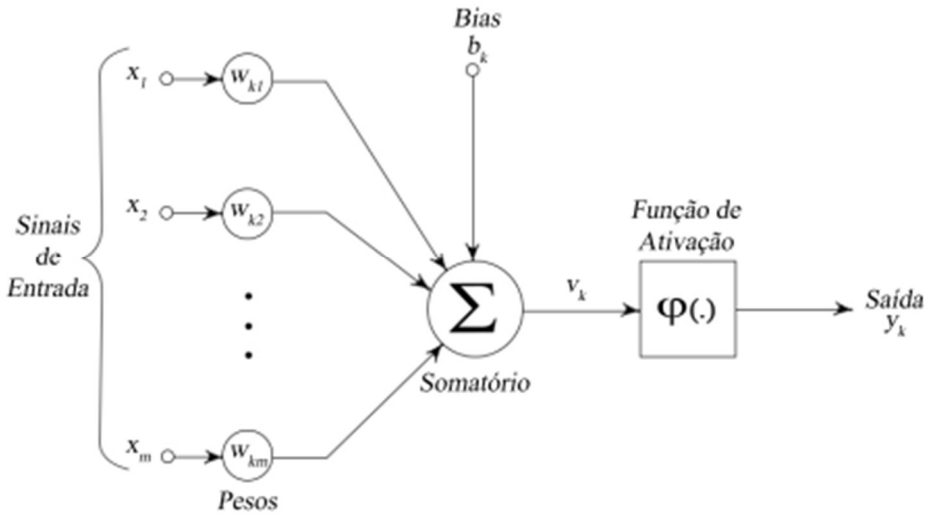
As RNAs são algoritmos computacionais que apresentam um modelo matemático similar a estrutura neural biológica, segundo Nunes (2003), seu objetivo é fornecer subsídios que auxiliem na interpretação de dados – armazenando o conhecimento e aplicando em situações reais. As RNAs se apresentam, então, como uma ferramenta estatística capaz de armazenar conhecimento por meio de exemplos – o algoritmo generaliza esses dados e memoriza o conhecimento dentro de parâmetros adaptáveis de rede (os pesos). Esse modelo permite ainda detectar

implicitamente relações não lineares entre a variável resposta e as explicativas atuando em situações como classificação de padrões, aproximação de funções, segmentação em classes e predição de séries temporais – que é o caso da presente pesquisa.

Do ponto de vista estrutural, as RNAs podem ser classificadas como redes estáticas – não recorrentes – ou recorrentes – dinâmicas e a principal diferença é a presença, ou não, de conexões que realimentam os neurônios da rede (HAYKIN, 2001). As Redes Neurais Artificiais apresentam ainda conexões interneurais, representadas pelos pesos sinápticos, que se concretizam como uma forma de armazenamento e processamento de conhecimento.

Apesar de ser considerado recente, o estudo de modelos computacionais que simulem o funcionamento de neurônios data os anos 40 com o estudo de McCulloch e Pitts (1943) – que interpretou o cérebro sob a ótica computacional. O neurônio é a unidade de processamento fundamental de uma Rede Neural Artificial e se baseia em três principais elementos: conjunto de sinapses – caracterizado por um peso –, um somador – que soma os sinais de entrada, ponderados pelas respectivas sinapses – e a função de ativação – que restringe a saída de um neurônio (HAYKIN, 2001), como exemplificado na Figura 2.

Figura 2 - Modelo não linear de um neurônio artificial



Fonte: Haykin (2001)

O modelo neuronal ainda inclui um *bias* que, aplicado externamente, tem o efeito de alterar a entrada na função de ativação. Ainda nesse contexto, de acordo

com Haykin (2001), o neurônio artificial se assemelha ao humano em dois aspectos básicos: o conhecimento é adquirido do ambiente – por meio de processos de aprendizagem; e os pesos sinápticos são utilizados para armazenar o conhecimento adquirido. Ou seja, o aprendizado em uma RNA pode ser caracterizado como um processo de otimização dos pesos sinápticos, *bias* – e o algoritmo de retropropagação pode ser utilizado no ajuste desses parâmetros de modo que a função de erro seja minimizada.

As RNAs são utilizadas na resolução de problemas complexos – ou seja, em situações em que não se há o rigoroso conhecimento sobre o comportamento das variáveis. No que tange a topologia, para implementar uma Rede Neural Artificial, segundo Santos *et al.* (2005), é necessária a definição de:

a) O número de nós na camada de entrada;
b) O número de camadas ocultas e seu número de neurônios;
c) O número de neurônios na camada de saída.

Ainda segundo Santos *et al.* (2005), esses parâmetros afetam o desempenho da RNA e devem ser criteriosamente escolhidos.

Além das diferentes composições estruturais possíveis, deve-se escolher a arquitetura que melhor desempenhe no problema escolhido. Nas RNAs, assim como em outras técnicas de *Machine Learning*, separa-se os dados em dois conjuntos, um chamado de treino (o maior conjunto, sendo em torno de 80% dos dados) e o conjunto de teste (o menor, 20% restante dos dados). O conjunto de treino é usado para que a técnica aprenda (fase de treinamento) com os padrões apresentados a ela. Já o conjunto de teste é usado para avaliar o desempenho da técnica.

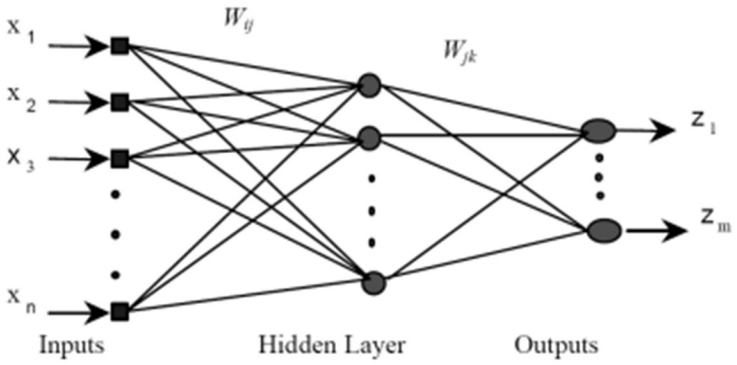
2.1.1 Multi Layer Perceptron (MLP)

Multi Layer Perceptron (MLP) ou Perceptron de múltiplas camadas no português é uma arquitetura de Rede Neural Artificial composta por 3 tipos de camadas: a de entrada, camadas intermediárias (ou ocultas) e a camada de saída – onde o sinal se transforma no resultado da rede apresentado na Figura 3. Esse resultado é comparado a uma resposta verdadeira, proveniente dos dados de treinamento, e, por meio dessa comparação, torna-se possível avaliar o erro, que segue o caminho inverso a rede e ajusta os pesos sinápticos, minimizando, assim, os sinais errôneos de saída. A aprendizagem dessa Rede Neural Artificial é pautada,

então, na alteração dos pesos sinápticos (HAYKIN, 2001) e a adição de uma ou mais camadas permite que a rede melhor mapeie os problemas.

Esse tipo de arquitetura é chamado de *feed-forward*: toda informação da rede segue em direção aos *Perceptrons* (neurônios) de saída. Sua arquitetura ainda garante que todo neurônio está conectado a todos os outros da camada anterior – ou seja, as camadas são totalmente conectadas, conforme observado na Figura 3.

Figura 3 - Arquitetura da Rede MLP



Fonte: Niroobakhsh *et al.* (2012)

Ressalta-se que uma das maiores preocupações relacionadas ao treinamento da Rede Neural Artificial é o ajuste excessivo nos dados de treinamento, ou seja, o modelo de predição desenvolvido funcionará bem para o problema especificado, mas terá, possivelmente, um mal desempenho na generalização para outros dados, problema conhecido como *Overfitting* – que será mais bem especificado em uma futura seção.

Na arquitetura de rede MLP o neurônio pode ser descrito pela Equação 1.

$$y = f\sum_{j=1}^{n}\left(w_j * x_j + \theta\right) = f(w^t * x + \theta) \qquad (1)$$

onde x é a entrada, w a conexão sináptica, n é o número de neurônios em uma camada específica e θ é o *bias* do neurônio. A função f é a função de ativação.

Como supracitado, o algoritmo de retropropagação pode ajustar esses parâmetros de forma a minimizar o erro, ajustando os pesos sinápticos e os *bias*.

.

2.1.2 Long Short Term Memory (LSTM)

Redes neurais recorrentes com memória de longo prazo, em inglês *Long Short Term Memory* (LSTM) emergiram como uma solução escalável e eficaz para problemas de aprendizagem com dados sequenciais (GREFF *et al.,* 2017). A arquitetura LSTM é usada para monitoramento de integridade de pontes, que é originalmente usado para lidar com questões envolvendo séries temporais, se caracterizando como uma rede melhorada derivada da Rede Neural Recorrente (RNN). Essa Rede Neural armazena sequências em sua memória interna por um longo tempo – fato que aumenta a precisão na tomada de decisão da próxima etapa quando comparada com as redes tradicionais *feedforward* (GUO *et al*., 2019). O modelo LSTM estende a memória da RNN, permitindo o aprendizado de dependências de entradas a longo prazo. A memória dessa arquitetura é conhecida como célula *gated* – inspirada na capacidade de tomar a decisão de preservar ou ignorar informações da memória – que é baseada nos valores de pesos atribuídos a informação durante o processo.

A célula é ainda coposta por 3 portões principais: portão de esquecimento (*forget gate*), portão de entrada (*input gate*) e o portão de saída (*output gate*), e cada um dos portões usa a função de ativação sigmoide para controlar o fluxo de informação.

O portão de esquecimento decide quais informações antigas devem ser mantidas ou esquecidas na célula, e é calculado usando a Equação 2.

$$f_t = \delta w_f * [h_{t-1}, X_t] + b_f \qquad (2)$$

- f_t é o vetor de ativação do portão de esquecimento no tempo t;
- δ é a função sigmoide;
- w_f é a matriz de pesos para o portão de esquecimento;
- h_{t-1} é o estado oculto da célula LSTM no tempo $t-1$;
- X_t é a entrada no tempo t;
- b_f é o vetor de *bias* para o portão de esquecimento.

O portão de entrada decide quais novas informações devem ser atualizadas na célula LSTM, conforme Equações 3 e 4. Ele também usa a função sigmoide e a operação de multiplicação elementar.

$$i_t = \delta(W_i * [h_{t-1}] + b_i) \qquad (3)$$

$$c_t = \tanh(W_c * [h_{t-1}, x_t] + b_c) \quad (4)$$

onde:

- i_t é o vetor de ativação do portão de entrada no tempo t;
- c_t é o vetor de candidato a atualização do estado da célula no tempo t;
- $tanh$ é a função tangente hiperbólica;
- W_i e W_c são as matrizes de pesos para o portão de entrada;
- b_i e b_c são os vetores de viés para o portão de entrada.

O portão de saída decide qual parte do estado da célula será enviada como saída da célula LSTM (Equação 5).

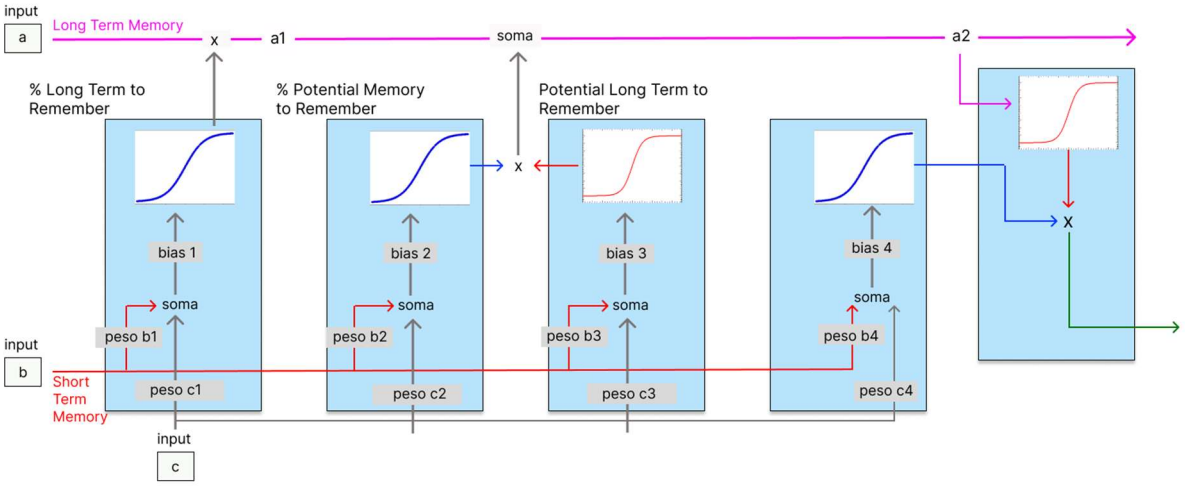
$$O_t = \delta(W_o * [h_{t-1}, x_t] + b_o) \quad (5)$$

- o_t é o vetor de ativação do portão de saída no tempo t;
- W_o é a matriz de pesos para o portão de saída;
- b_o é o vetor *bias* para o portão de saída.

A rede neural LSTM pode, então, ser utilizada para resolver problemas de desaparecimento e explosão de gradiente (tendência a zero ou ao infinito) por meio da sua arquitetura múltipla – que se concretiza por meio de duas funções de ativação: sigmoide (que varia de 0 a 1) e tangente hiperbólica (que varia de -1 a 1). Sua arquitetura também divide eventos que aconteceram há algum tempo (termo de longa memória ou, no inglês, *long term memory*) e eventos que acabaram de ocorrer (termo de curta memória ou *short term memory*), possibilitando dar pesos diferentes para cada um dos eventos observados.

No que tange ao funcionamento de fato, a rede neural se inicia, como mostrado na Figura 4, multiplicando o *input* (*c*) e o termo de curta memória (*b*) pelos seus respectivos pesos. A soma dos resultados – somado, ou não, a um *bias* (*bias* 1) – vira *input* da função sigmoide e o valor é, posteriormente, multiplicado pelo termo de longa memória (*a*) gerando *a1*. A segunda etapa se inicia, também, com a multiplicação do *input* (c) e termo de curta memória (*b*) pelos respectivos pesos e a soma de transforma em *input* da função sigmoide e tangente hiperbólica e, para essa etapa, a multiplicação dos dois valores é somado ao termo de longa memória obtido anteriormente (*a1*), gerando *a2*.

Figura 4 - Arquitetura da Rede LSTM



Fonte: a autora (2023) adaptado de Olah (2015)

A ideia central da arquitetura LSTM é uma memória celular, que pode manter seu estado ao longo do tempo (GREFF *et al.*, 2017).

2.1.3 Funções de ativação

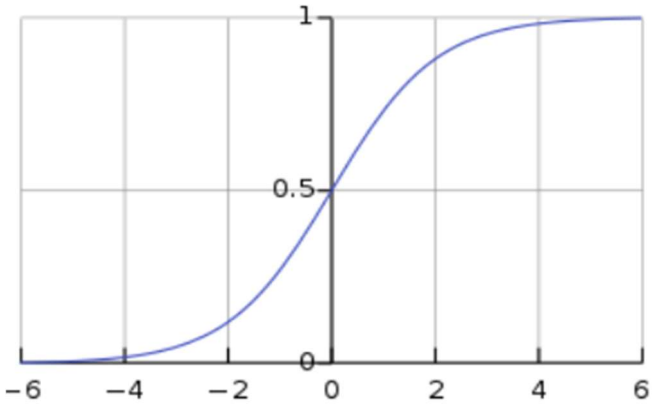
A função de ativação possibilita a transformação não linear do *input* - uma Rede Neural Artificial sem função de ativação se pauta apenas em uma equação linear, com limitada capacidade de resolução de problemas e baseada apenas em um modelo de regressão. Dessa forma, a função de ativação é um componente matemático agregado na estrutura neuronal que permite que o modelo execute atividades de maior complexidade e atue na resolução de problemas mais específicos.

A função de ativação é, então, aplicada em cada neurônio da camada oculta, após a soma ponderada dos sinais de entrada. Essa função é responsável por determinar a saída do neurônio e propagá-la para a próxima camada da rede e, para isso, algumas funções se destacam, tais como:

a) Função binária: função que ativa ou não um neurônio. Tem seu uso pautado principalmente em classificação de pertencimento, ou não, de uma determinada classe.
b) Função Sigmoide: função frequentemente utilizada em RNAs devido a sua saída, que está sempre no intervalo entre 0 e 1, o que é útil para a

classificação binária de dados demonstrados na Figura 5. No que tange a sua limitação, essa função pode levar a problemas de saturação, que ocorrem quando a saída do neurônio fica próxima dos extremos da curva, dificultando a atualização dos pesos.

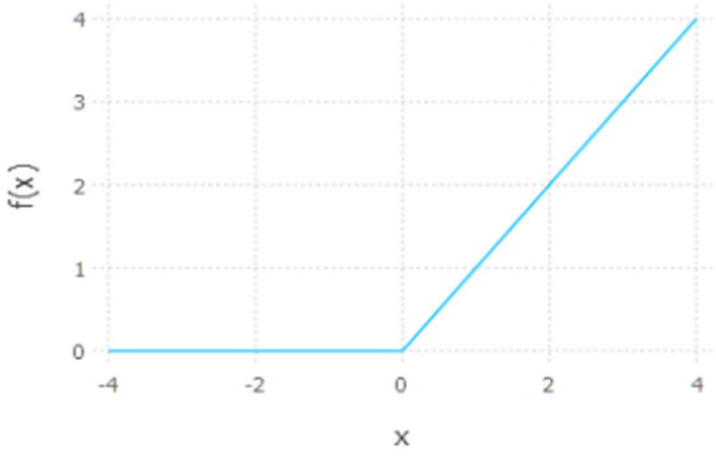
Figura 5 - Representação gráfica da função sigmoide



Fonte: Gharat (2019)

a) Função ReLU: *Rectified Linear Unit* é uma função simples que retorna a entrada do neurônio, caso seja positiva, e 0, caso contrário demonstrados na Figura 6. A função ReLU tem a vantagem de ser fácil de calcular e não apresentar problemas de saturação. Sobre as desvantagens, ressalta-se os problemas de "neurônios mortos", que ocorrem quando a saída do neurônio é 0 para todas as entradas, impedindo a atualização dos pesos.

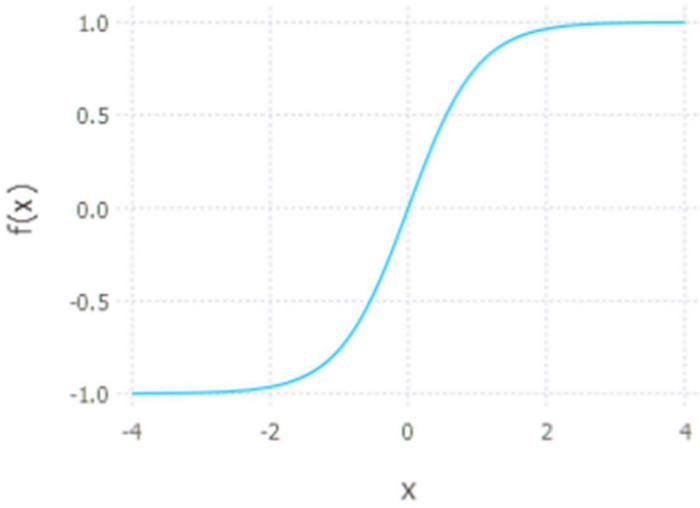
Figura 6 - Representação da Função ReLU



Fonte: Gharat (2019).

b) Função Tanh: tangente hiperbólica é uma função sigmoide que retorna a saída no intervalo entre -1 e 1 (Figura 7). A função Tanh é útil para a classificação binária de dados, mas apresenta problemas de saturação, assim como a função sigmoide.

Figura 7 - Representação gráfica da função Tanh



Fonte: Gharat (2019).

c) Função *Softmax*: frequentemente utilizada na camada de saída de RNAs que realizam classificação multiclasse. Ela normaliza as saídas dos neurônios de forma que a soma das saídas seja igual a 1. Isso permite que as saídas possam ser interpretadas como probabilidades de pertencer a cada classe.

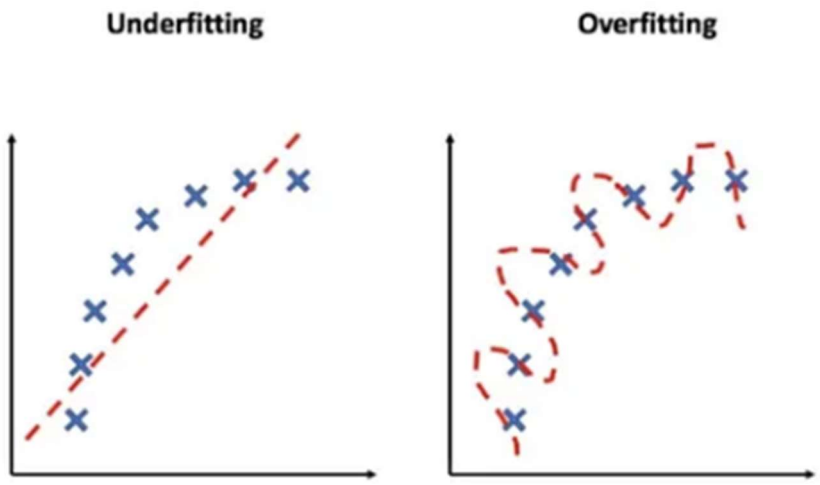
A escolha da função de ativação em uma Rede Neural Artificial depende das características dos dados e do tipo de problema que está sendo resolvido. Essa etapa não é uma ciência exata e pode depender de testes empíricos e validação cruzada para determinar a função que melhor se adequa ao problema em questão. Além disso, outras técnicas de ajuste de hiperparâmetros, como a variação do número de camadas e neurônios da rede, podem ser importantes para obter uma RNA com bom desempenho na previsão do preço do petróleo.

2.1.4 Overfitting e Underfitting

No que tange a esfera de *Machine Learning*, um modelo capaz de se ajustar aos dados de treinamento não necessariamente terá um bom desempenho nos dados de teste e essa disparidade é chamada de *Gap* de Generalização. Dentre as principais dificuldades encontradas no momento de treinamento de redes neurais artificiais, destaca-se o sobreajuste (*overfitting*) – onde o modelo apresenta baixa performance no conjunto de testes e alta no conjunto de treino - e o subajuste (*underfitting*) – onde o modelo apresenta baixa performance no conjunto de testes e treino, como exemplificado na Figura 8.

.

Figura 8 - Processos de Overfitting e Underfitting



Fonte: Addagatla (2021)

Underfitting ocorre, então, quando o modelo não é capaz e obter um valor de erro suficientemente baixo no treinamento (ZHANG *et al*., 2019), e isso pode acontecer quando a RNA é muito pequena, com poucas camadas ou poucos neurônios em cada camada. Nesse caso, a RNA não é capaz de aprender os padrões importantes nos dados de treinamento, resultando em uma perda de precisão na previsão de novos dados.

O fenômeno *overfitting* pode correr em situações em que os dados de treinamento não representam de fato o conjunto dos possíveis dados de entrada do modelo. Outras possíveis causas são modelos com alta variância ou modelos muito

complexos. Para evitar-se o *overfitting* faz-se necessário técnicas de regularização, redução do tamanho da rede e redução da taxa de aprendizagem. Além disso, pode interromper-se o treinamento em um estágio anterior (GOODFELLOW *et al*., 2016), fato que pode fazer com que o modelo não consiga aprender suficientemente os dados – dificultando a captura da tendência dominante, ocorrendo o *underfitting*.

Fica claro, portanto, que a prevenção de *overfitting* e *underfitting* ajuda no melhoramento da performance e desempenho do aprendizado da máquina. É importante escolher uma RNA que seja suficientemente complexa para capturar os padrões importantes nos dados de treinamento, mas não tão complexa que memorize tais dados.

2.1.5 Medidas de erro

Para avaliação das previsões realizadas, algumas medidas de erros são comumente utilizadas como:

a) Erro quadrático médio (*Mean Squared Error* – MSE): fornece uma medida do desvio médio quadrático entre as previsões e os valores reais. O MSE penaliza erros grandes mais do que erros menores devido ao uso do quadrado das diferenças. Sua definição é dada pela Equação 6.

$$MSE = \frac{1}{n}\sum_{k=1}^{n}(y_t - \hat{y_t})^2 \qquad (6)$$

- n é o número total de amostras ou observações;
- y_t representa o valor real na amostra t;
- $\hat{y}_t$ representa o valor previsto ou estimado na amostra t.

b) Erro médio absoluto (*Mean Absoute Error* – MAE): é a média das diferenças absolutas entre e previsão e os valores reais, mensurando o desvio médio. Sua fórmula é calculada com a diferença entre cada valor real e previsão (Equação 7).

$$MAE = \frac{1}{n}\sum_{k=1}^{n}(|y_t - \hat{y}_t\,|) \qquad (7)$$

2.2 SÉRIES TEMPORAIS

Séries temporais são uma forma de organizar dados quantitativos ao longo do tempo, possibilitando análises e predição. Analisar séries temporais se torna um importante fator para verificação de padrões e tendências importantes e criar previsões de movimentos futuros que podem, então, pautar tomada de decisões. A seguir, serão abordadas algumas características de séries temporais.

A estacionariedade é uma propriedade desejável em séries temporais – visto que facilita a modelagem e previsão. Séries estacionárias possuem média constante ao longo do tempo, não existindo tendências de alta ou de baixa, ou seja, a estacionariedade garante que propriedades estatísticas sejam mantidas constantes ao longo do tempo.

Uma série temporal é estacionária quando ela se desenvolve aleatoriamente, no tempo, em torno de uma média constante, refletindo alguma forma de equilíbrio estável (BEZERRA, 2006).

Para iniciar um modelo de predição, faz-se necessário a verificação das propriedades estáticas dos dados, verificando, ou não, a estacionariedade. Uma forma de verificar essa característica é analisar a média dos dados: médias constantes implicam estacionariedade. Há também testes estatísticos disponíveis que verificam essa característica em um conjunto de dados fornecidos.

Séries temporais podem apresentar diversos componentes que armazenam dados históricos que auxiliam no entendimento do seu comportamento passado e futuro. Esses componentes são, segundo Morettin e Toloi (2004): tendência - movimento evolutivo de crescimento ou decrescimento; sazonalidade - padrões de oscilação que se repetem em tempos uniformes; ciclicidade - padrões de oscilação que se repetem em tempos não uniformes, geralmente superiores a um ano; e aleatoriedade - padrão de oscilação aleatória.

Os componentes de uma série temporal permitem entender melhor o comportamento dos dados e podem ajudar a melhorar as previsões futuras.

3 METODOLOGIA

A presente pesquisa, de abordagem quantitativa, possui como procedimento metodológico o *Design Science Research* (DSR), que é classificado como um método condutor de pesquisas de cunho tecnológico, como, por exemplo, desenvolvimento de Redes Neurais Artificiais, constituindo-se em uma abordagem que, quando bem aplicada, produz rigor científico efetivo (LACERDA, 2013). A metodologia DSR é, ainda, considerada um processo com ênfase na criação de conhecimento prático e soluções utilizáveis, visando não só a produção de conhecimento teórico como também a geração de artefatos que possam ser implementados e utilizados para resolução de problemas reais.

Visando orientar e padronizar a resolução de problemáticas, Wieringa (2009) propõe um ciclo regulador (Figura 9), que se inicia com a investigação do problema.

Figura 9 - Ciclo Regulador DSR

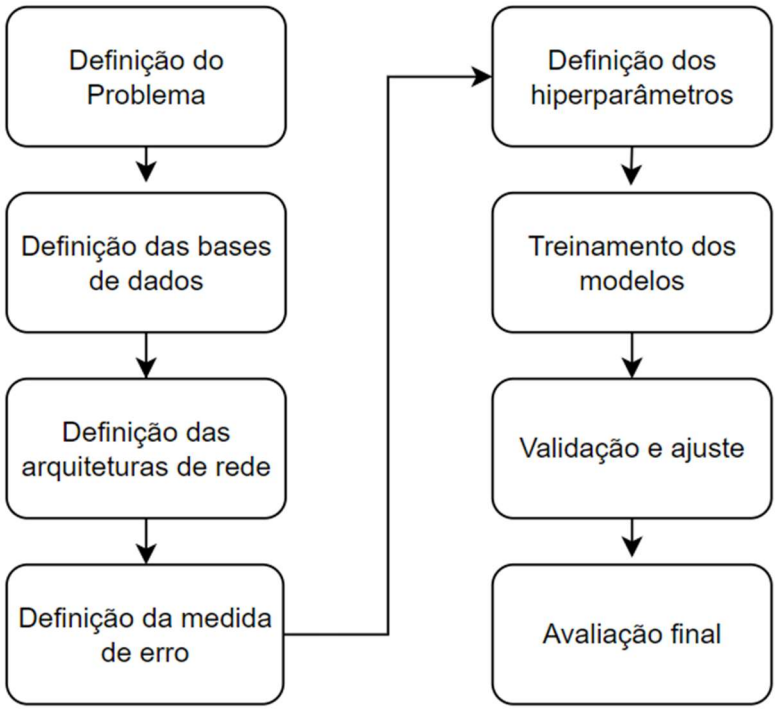
Fonte: a autora (2023) adaptado de Wieringa (2009)

A etapa seguinte se baseia no projeto de soluções, momento de definição de estratégias que podem solucionar o empasse definido na primeira fase do processo. A validação do projeto, terceira etapa do ciclo regulador, se baseia na construção de conhecimento, ou seja, nessa etapa o pesquisador analisa e pondera os potenciais resultados da implementação bem-sucedida do projeto idealizado. A quarta etapa se refere a implementação, processo totalmente prático, que é seguido pela quinta e

última etapa do ciclo: avaliação da implementação, que tem como objetivo gerar conhecimento científico a partir da pesquisa realizada. Fica nítido, então, a relevância de se ter como procedimento metodológico da presente pesquisa a DSR, que tem como função amparar o pesquisador e ajudá-lo no desenvolvimento do artefato.

Seguindo o ciclo regulador de Wieringa (2009) e adaptando para as necessidades da presente pesquisa, criou-se o diagrama a seguir, que visa mapear os processos para realização do estudo (Figura 10).

Figura 10 - Fluxograma metodológico da pesquisa



Fonte: a autora (2023)

A etapa de definição do problema se deu quando a pesquisadora se aprofundou na temática petróleo e gás, entendendo-se a importância e a relevância do preço do petróleo para inúmeras esferas econômicas e estratégicas da sociedade. O petróleo é uma *commodity* global que influencia diretamente os setores de energia, transporte e abastecimento, manufatura e outros, e a flutuação de seu preço tem impacto significativo. Assim, a capacidade de antecipar mudanças no preço dessa *commodity* é fundamental para embasamento de decisões estratégicas, mitigando riscos associados. No que tange a definição de base de dados, procurou-se em fontes como

Kaggle que abrangessem uma quantidade significativa de anos com os preços do barril de petróleo (ao menos 10 anos de informações). Os dados coletados do petróleo variam de março de 1983 até junho de 2023, com dados mensais. Adicionalmente, para a análise multivariada (explicada a seguir), utilizou-se uma base de dados com os valores médios mensais da cotação do dólar em relação ao real, que foi obtida usando uma função no *software* R. Extraiu-se os resultados em formato csv para que fosse, posteriormente, lido em Python.

Para a etapa seguinte, definição da arquitetura das redes, utilizou-se dois critérios de seleção: arquitetura de rede mais utilizada na revisão de literatura e arquitetura de rede mais simples. Assim, torna-se possível a comparação entre dois métodos. Para a presente pesquisa, selecionou-se, então, a LSTM e MLP, respectivamente. Já para a seleção do critério de avaliação do erro, 4ª etapa, optou-se por empregar a medida mais utilizada nos artigos encontrados: erro quadrático médio (Equação 6), que é uma métrica comumente utilizada em modelos para avaliação de predição de séries temporais. Dentre as vantagens da utilização dessa medida de erro, destaca-se a sensibilidade aos desvios, penalizando de forma quadrática a diferença entre os valores, e a facilidade de interpretação dos valores apresentados, que são intuitivos, MSE mais baixo significa uma melhor adequação do modelo aos dados observados. Para a definição da melhor arquitetura de rede, escolherá a que obtiver o menor erro.

Para a rede neural MLP, o número de neurônios na camada de entrada será testado com 2 a 6 neurônios para a análise univariada – ou seja, de 2 a 6 informações passadas do preço do barril de petróleo. O número de neurônios nas camadas ocultas será variado de 1 a 8, o número de camadas foi variado manualmente, testando uma e duas camadas, e o número de neurônio na camada de saída será 1 (preço do barril de petróleo a ser previsto um período à frente). Os valores utilizados para taxa de aprendizado foram 0,001 e 0,01 e a função de ativação foi fixada como relu.

Já em uma rede LSTM, os principais parâmetros que podem ser variados, além da taxa de aprendizado e tamanho da janela, que foram previamente explicitados, são:

a) Número de unidades LSTM: O número de unidades LSTM define a capacidade da camada oculta da rede LSTM. Um maior número de unidades pode ajudar a capturar relações mais complexas nos dados, mas também aumenta a complexidade do modelo. No presente estudo, variou-se o número

de unidades LSTM manualmente, e os valores testados foram 1, 10, 20, 30, 40, 50, 60, 70, 80 e 90.

b) Número de camadas LSTM: É possível empilhar várias camadas LSTM para formar uma rede LSTM profunda. Cada camada adicional permite que o modelo aprenda representações hierárquicas dos dados, mas também aumenta a complexidade e o tempo de treinamento do modelo. Para o presente trabalho, fixou-se apenas uma camada, já que seriam avaliados apenas os dados históricos da série temporal e não havia necessidade de aumentar a complexidade do modelo.
c) Função de ativação: Assim como em MLP, é possível escolher diferentes funções de ativação para as unidades LSTM, como tangente hiperbólica ou função sigmoide. A função de ativação determina a saída das unidades LSTM e influencia a capacidade de aprendizado e a propagação do gradiente durante o treinamento. Para o presente estudo, fixou-se a função de ativação Tangente Hiperbólica.
d) Todo o trabalho será realizado no *software Visual Studio*, utilizando a biblioteca *keras*.

3.1 DESENVOLVIMENTO DA ARQUITETURA *MULTI LAYER PERCEPTRON*

A primeira etapa para o desenvolvimento da arquitetura MLP é a definição da função, onde o modelo será criado (Figura 11). Para isso, passam-se como parâmetros alguns argumentos como: taxa de aprendizado (*learning_rate*), número de neurônios das camadas ocultas – para o caso da presente pesquisa, como citou-se anteriormente, testou-se o uso de uma ou duas camadas ocultas – e o número dos dados de entrada (*input_shape).* O modelo da RNA é criado, então, como um objeto *Sequential* – que garante que as camadas sejam construídas de forma sequencial. E as camadas são definidas posteriormente usando o método "*model.add"* – que é uma forma usada principalmente em bibliotecas de *deep learning* para adicionar camadas à arquitetura de uma RNA. No contexto da presente pesquisa, "*model"* é uma instância de um modelo sequencial utilizado para alimentar a MLP. O modelo é então compilado com a função "*compile",* que tem como principais argumentos a função de perda, que, no caso do presente estudo foi definido a MSE, e o otimizador com a taxa de

aprendizado, optou-se por utilizar o otimizador Adam – que é um método de otimização que calcula as taxas de aprendizado adaptativas para cada parâmetro. Adam mantem uma média exponencial decrescente dos gradientes anteriores (RIBEIRO; ARAÚJO-JUNIOR, 2020).

Figura 11 - Definindo a função para criação do modelo MLP

```
# Definir uma função para criar o modelo MLP
def create_model(learning_rate, num_neurons_1, num_neurons_2, input_shape):
    model = Sequential()
    model.add(Dense(num_neurons_1, activation='relu', input_shape=(input_shape,)))
    model.add(Dense(num_neurons_2, activation='relu'))
    model.add(Dense(1))

    optimizer = Adam(learning_rate=learning_rate)
    model.compile(loss='mean_squared_error', optimizer=optimizer, metrics=['mse'])

    return model
```

Fonte: a autora (2023).

O passo seguinte se baseia na criação de uma *DataFrame* para alocação dos resultados (Figura 12). E também na criação de uma função que cria uma sequência de dados a partir de uma série temporal (Figura 13). Os parâmetros da função são o próprio conjunto de dados (do inglês, *data)* o qual deseja-se criar sequências, e o número (inteiro) que definirá o tamanho das sequências (do inglês, *time_steps)*. Cria-se, então, uma lista vazia que será utilizada para armazenar as sequências então criadas. E, usando o *loop* "*for"* para percorrer os dados originais e criar sequências a partir deles. O *loop* é executado do índice zero até o índice x (Equação 8), garantindo que a última sequência se encaixe completamente dentro dos dados originais. O valor de x é calculado subtraindo o valor do tamanho da sequência do tamanho total da variável *data.*

$$x = len(data) - time\_steps \qquad (8)$$

Dentro do *loop,* é criada a variável "seq", que armazenará a sequência, e o valor inicial dessa variável é o fatiamento dos dados originais. Em cada iteração, uma nova sequência é criada, deslizando os *time_steps* – deslizando-se pelos dados originais. A função retorna, então, uma lista de sequências, que contém todas as janelas deslizantes que serão utilizadas para treinamento e testagem do modelo.

Figura 12 - Criação da *DataFrame*

```
results_df = pd.DataFrame(columns=[
    'Window_Size',
    'Learning_Rate',
    'Neurons_Layer1',
    'Neurons_Layer2',
    'Best_MSE'
])
```

Fonte: a autora (2023).

Em resumo, a função *create_sequences* permite transformar o conjunto de dados em sequências de tamanho específico.

Figura 13 - Definição da função para criar sequências

```
# Função para criar sequências de dados
def create_sequences(data, time_steps):
    sequences = []
    for i in range(len(data) - time_steps + 1):
        seq = data[i:i + time_steps]
        sequences.append(seq)
    return sequences
```

Fonte: a autora (2023).

Inicia-se, então, a busca pelos melhores hiperparâmetros que possam compor a RNA MLP. Define-se uma lista de valores para janelas de tempo a serem testadas (*window_sizes),* cada tamanho de janela representa a quantidade de pontos consecutivos que serão utilizados como valores de entrada para a previsão do valor seguinte na série temporal. A etapa seguinte consiste em criar um *loop* para iterar nos diferentes tamanhos de janela. Dentro do *loop,* cria-se uma variável que armazena as sequências criadas a partir dos dados de preços. Cada tamanho de janela especificado é usado para criar sequências – por meio da função "*create_sequences"*. Converte-se então as sequências de dados em uma matriz *NumPy,* com o nome de "*data" (*do inglês, dados), que será utilizada para o treinamento do modelo MLP.

Divide-se, então, a base de dados em subconjuntos para treino e teste com o método *Holdout*. Nessa etapa, fatiou-se a *DataFrame* para que 80% dos dados representassem os dados para treinamento e 20% para teste. Os dados de

treinamento e teste são ajustados para criar as entradas (*x_train* e *x_test*) e saídas esperadas (*y_train* e *y_test*) para o modelo.

Segue-se com a definição das faixas de valores para otimização de hiperparâmetros. Varia-se, por meio de um *loop* a taxa de aprendizado, o número de neurônios na primeira camada oculta e o número de neurônios na segunda camada oculta, e combinam-se os parâmetros para achar a melhor combinação de valores. O modelo é treinado usando os dados de treinamento e o histórico é armazenado em "*history*". O modelo é então avaliado com os dados de teste e o erro quadrático médio é calculado. Se o MSE obtido foi o melhor até o momento, os valores são atualizados na *DataFrame* antes criada, com os valores do tamanho da janela, taxa de aprendizagem, número de neurônios, número de camadas ocultas e MSE (Figura 14).

Figura 14 - Busca pela melhor combinação de parâmetros MLP

```
window_sizes = [2,3,4,5,6]
for window_size in window_sizes:
    print("Testing window size:", window_size)
    sequences = create_sequences(dados['price'], window_size)
    data = np.vstack(sequences)
    train_size = int(0.8 * len(sequences))
    train_data = data[:train_size, :]
    test_data = data[train_size:, :]
    x_train = train_data[:, :window_size - 1]
    y_train = train_data[:, window_size - 1]
    x_test = test_data[:, :window_size - 1]
    y_test = test_data[:, window_size - 1]
    learning_rates = [0.001, 0.01]
    num_neurons_1_values = [1,2,3,4,5,6,7,8]
    num_neurons_2_values = [1,2,3,4,5,6,7,8]

    for lr in learning_rates:
        for n1 in num_neurons_1_values:
            for n2 in num_neurons_2_values:
                model = create_model(lr, n1, n2, input_shape=window_size - 1)
                history = model.fit(
                    x=x_train,
                    y=y_train,
                    epochs=100,
                    batch_size=32,
                    validation_split=0.1,
                    verbose=0
                )
                evaluation = model.evaluate(x_test, y_test)
                mse = evaluation[0]
                if mse < best_mse:
                    best_mse = mse
                    best_hyperparameters = {
                        'Learning_Rate': lr,
                        'Neurons_Layer1': n1,
                        'Neurons_Layer2': n2
                    }
                    best_window_size = window_size
                    results_df = pd.DataFrame({
                        'Window_Size': [best_window_size],
                        'Learning_Rate': [best_hyperparameters['Learning_Rate']],
                        'Neurons_Layer1': [best_hyperparameters['Neurons_Layer1']],
                        'Neurons_Layer2': [best_hyperparameters['Neurons_Layer2']],
                        'Best_MSE': [best_mse]
                    })
```

Fonte: a autora (2023).

Cria-se então um segundo modelo (“*best_model2”),* que é treinado utilizando os melhores hiperparâmetros anteriormente encontrados. O treinamento ocorre ao longo de 100 épocas e com lotes de tamanho 32.

Este trecho de código efetivamente cria um modelo com os melhores hiperparâmetros encontrados durante a busca e treina esse modelo com os dados de treinamento correspondentes. O modelo resultante, *best_model2*, é otimizado para a tarefa de previsão com base na configuração de hiperparâmetros que levou ao menor MSE durante a busca.

Para incrementar a análise, resolveu-se testar a adição do preço do dólar em relação ao real como *input* da RNA, a fim de constatar, ou não, uma melhora na performance do modelo ao se ter uma análise além da própria série temporal (chamada neste estudo de análise multivariada).

Criou-se então uma nova função que estruturasse um *DataFrame* de forma adequada para ser passado como *input* da RNA (Figura 15).

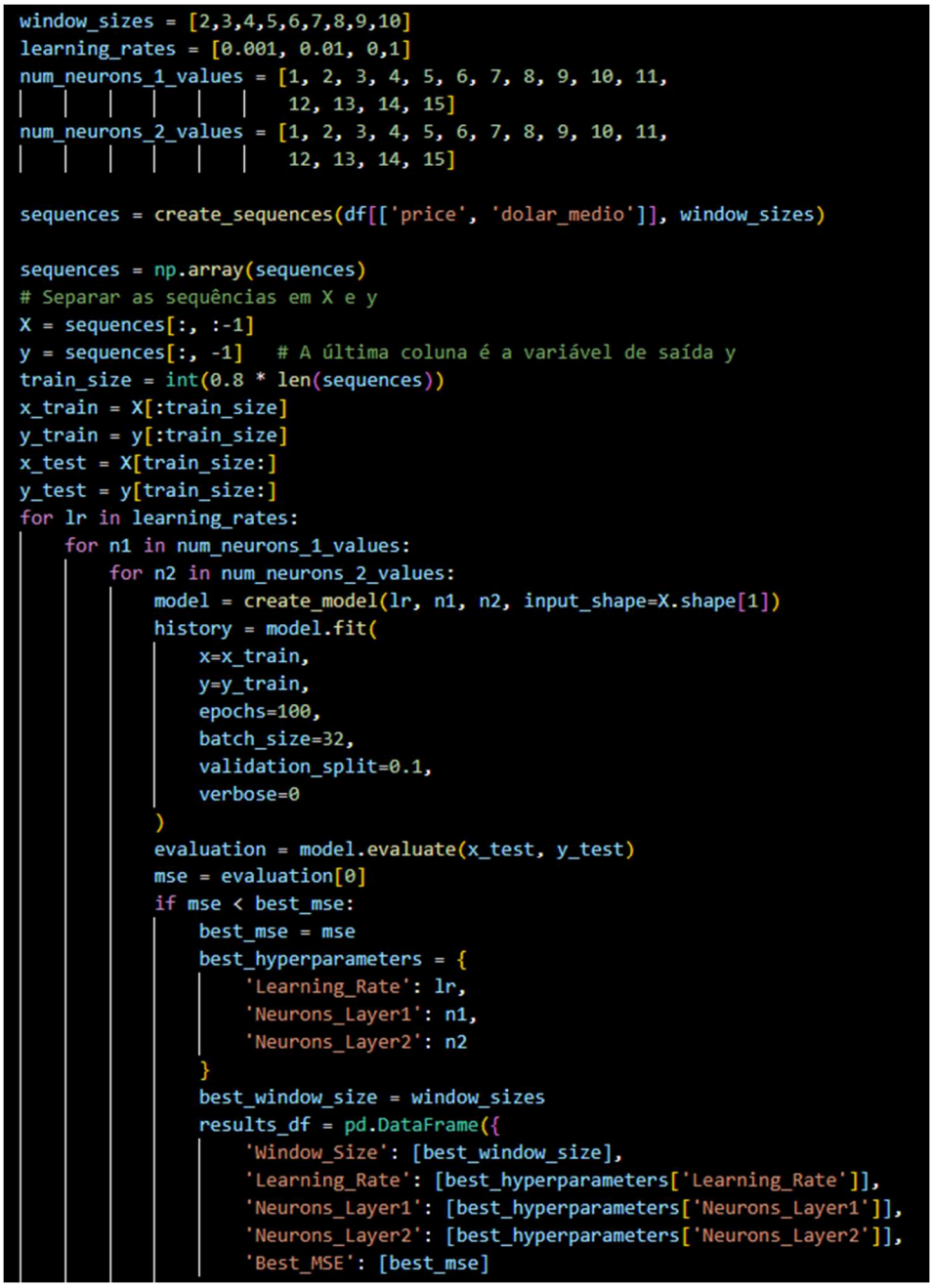
Figura 15 - Função para criação da base de dados a ser passada na análise multivariada

```
def create_lagged_df(df, window_size):
    df_dict = {}
    for i in range(1, window_size + 1):
        df['crude_oil_lag_' + str(i)] = df['price'].shift(i)
        df['dollar_lag_' + str(i)] = df['dolar_medio'].shift(i)
    df = df.dropna()
    df_dict[i] = (df.drop(columns=['price']), df['price'])
    return df_dict
```

Fonte: a autora (2023)

E adicionou-se a função criada no *looping* já existente. Para a análise multivariada, testou-se como tamanho da janela os valores de 2 a 10, taxa de aprendizado sendo 0,001 ou 0,01 ou 0,1 e número de neurônios nas camadas ocultas variando de 1 a 15, conforme pode ser observado na Figura 16. Testou-se com uma e duas camadas ocultas e a função de ativação, assim como na análise univariada, foi a relu.

Figura 16 - Criação da arquitetura MLP passando o valor do dólar como *input*

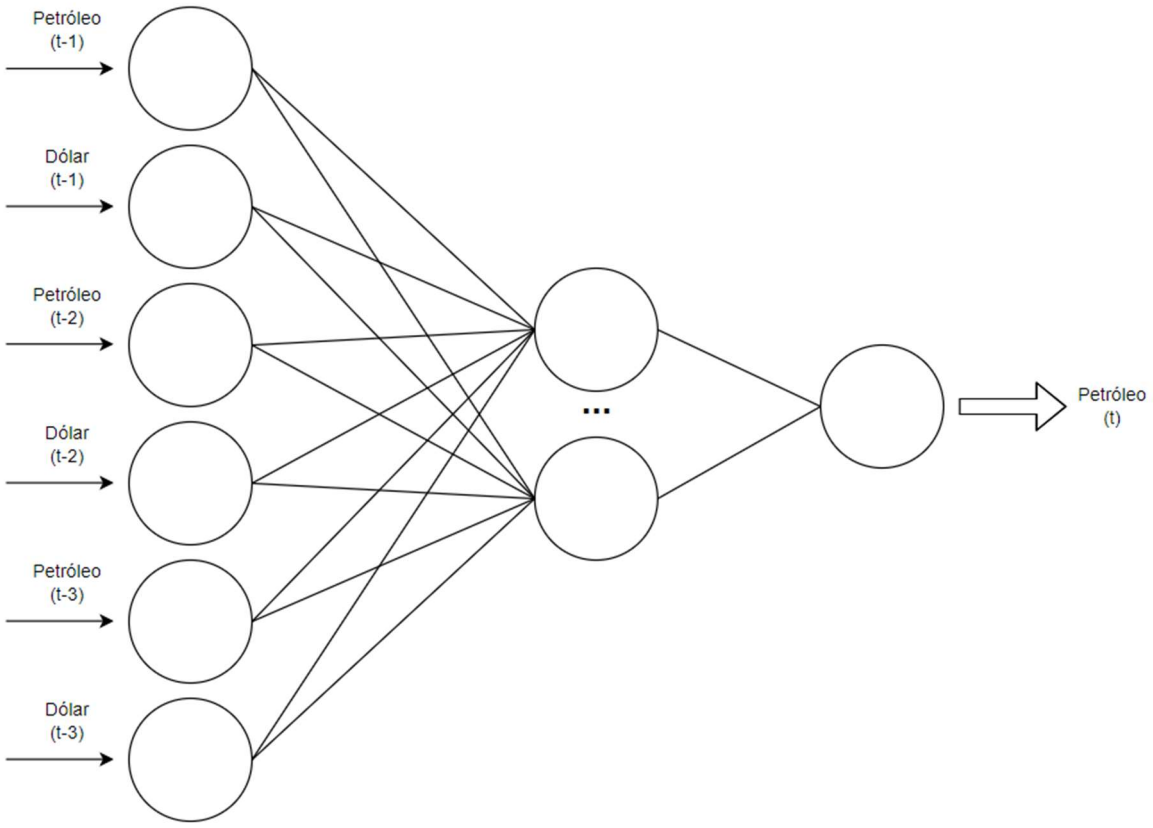


```
window_sizes = [2,3,4,5,6,7,8,9,10]
learning_rates = [0.001, 0.01, 0,1]
num_neurons_1_values = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
                        12, 13, 14, 15]
num_neurons_2_values = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
                        12, 13, 14, 15]

sequences = create_sequences(df[['price', 'dolar_medio']], window_sizes)

sequences = np.array(sequences)
# Separar as sequências em X e y
X = sequences[:, :-1]
y = sequences[:, -1]   # A última coluna é a variável de saída y
train_size = int(0.8 * len(sequences))
x_train = X[:train_size]
y_train = y[:train_size]
x_test = X[train_size:]
y_test = y[train_size:]
for lr in learning_rates:
    for n1 in num_neurons_1_values:
        for n2 in num_neurons_2_values:
            model = create_model(lr, n1, n2, input_shape=X.shape[1])
            history = model.fit(
                x=x_train,
                y=y_train,
                epochs=100,
                batch_size=32,
                validation_split=0.1,
                verbose=0
            )
            evaluation = model.evaluate(x_test, y_test)
            mse = evaluation[0]
            if mse < best_mse:
                best_mse = mse
                best_hyperparameters = {
                    'Learning_Rate': lr,
                    'Neurons_Layer1': n1,
                    'Neurons_Layer2': n2
                }
                best_window_size = window_sizes
                results_df = pd.DataFrame({
                    'Window_Size': [best_window_size],
                    'Learning_Rate': [best_hyperparameters['Learning_Rate']],
                    'Neurons_Layer1': [best_hyperparameters['Neurons_Layer1']],
                    'Neurons_Layer2': [best_hyperparameters['Neurons_Layer2']],
                    'Best_MSE': [best_mse]
```

Fonte: a autora (2023)

A Figura 17 ilustra a configuração da rede MLP exemplificada que será testada no exemplo multivariado. No exemplo, há 6 neurônios na camada de entrada, sendo 3 deles o preço do barril de petróleo nos últimos 3 meses (a variável t representa o período atual), e os outros 3 o valor da cotação do dólar.

Figura 17 - Ideia de rede MLP a ser testada



Fonte: a autora (2023).

3.2 DESENVOLVIMENTO DA ARQUITETURA *LONG SHORT TERM MEMORY*

Assim como fez-se para o modelo MLP, criou-se um laço *for* para a busca pela melhor combinação de hiperparâmetros para a arquitetura LSTM. Definiu-se taxas de aprendizado (0,001 e 0,01) e *window_sizes* (2,3,4,5,10,15). Para a arquitetura LSTM, deve-se fazer a normalização dos dados, para isso, utilizou-se a função *MinMaxScaler* e escalou todos os dados de preço de 0 a 1 (Figura 18).

Figura 18 - Normalização dos dados

```
# Pré-processamento dos dados
scaler = MinMaxScaler(feature_range=(0, 1))
df['preco_norm'] = scaler.fit_transform(df['price'].values.reshape(-1, 1))
```

Fonte: a autora (2023).

Também foi necessária a criação de uma função para preparação dos dados, similar a função da MLP, "*prepare_data"* tem como objetivo criar pares de sequências de entradas e saídas a partir dos dados normalizados e com base do tamanho da janela anteriormente especificada (Figura 19).

Figura 19 - Preparação dos dados para treinamento da rede LSTM

```
# Preparação dos dados de treinamento
def prepare_data(data, window_size):
    X, Y = [], []
    for i in range(len(data) - window_size):
        X.append(data[i:i+window_size])
        Y.append(data[i+window_size])
    return np.array(X), np.array(Y)

X_train, Y_train = prepare_data(dados_treinamento['preco_norm'].values, window_size)
X_test, Y_test = prepare_data(dados_teste['preco_norm'].values, window_size)
```

Fonte: a autora (2023).

A etapa seguinte é a criação do modelo LSTM. Define-se uma classe "*LSTMModel*" que herda de "*nn.Module*" a criação do modelo LSTM. O modelo consiste em uma camada LSTM, uma camada linear e o método *foward,* que especifica como os dados fluem pela rede (Figura 20).

Figura 20 - Criação do modelo LSTM

```
# Criação do modelo LSTM
class LSTMModel(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(LSTMModel, self).__init__()  # Correção aqui
        self.hidden_size = hidden_size
        self.lstm = nn.LSTM(input_size, hidden_size, batch_first=True)
        self.fc = nn.Linear(hidden_size, output_size)

    def forward(self, x):
        output, _ = self.lstm(x)
        output = self.fc(output[:, -1, :])
        return output
```

Fonte: a autora (2023).

Avalia-se, então, o modelo e após cada época o MSE é calculado para medir o desempenho da rede. Se o MSE nos dados de teste for menor do que o melhor MSE

registrado até agora (*best_loss*), os melhores hiperparâmetros (*best_params*), o modelo (*best_model*), e a melhor perda (*best_loss*) são atualizados (Figura 21).

Figura 21 - Atualização dos Hiperparâmetros da rede LSTM

```
model = LSTMModel(1, 50, 1)
criterion = nn.MSELoss()
optimizer = Adam(model.parameters(), lr=lr)

# Treinamento do modelo
X_train = torch.Tensor(X_train).unsqueeze(-1)
Y_train = torch.Tensor(Y_train).unsqueeze(-1)
X_test = torch.Tensor(X_test).unsqueeze(-1)
Y_test = torch.Tensor(Y_test).unsqueeze(-1)

epochs = 70
batch_size = 32

for epoch in range(epochs):
    for i in range(0, len(X_train), batch_size):
        batch_X = X_train[i:i+batch_size]
        batch_Y = Y_train[i:i+batch_size]

        optimizer.zero_grad()
        output = model(batch_X)
        loss = criterion(output, batch_Y)
        loss.backward()
        optimizer.step()

    # Avaliação do modelo
    with torch.no_grad():
        train_predict = model(X_train)
        train_loss = criterion(train_predict, Y_train)
        test_predict = model(X_test)
        test_loss = criterion(test_predict, Y_test)

if test_loss < best_loss:
    best_loss = test_loss
    best_params = params
    best_model = model
```

Fonte: a autora (2023).

E após a busca pelos melhores hiperparâmetros, o melhor modelo é recriado com esses hiperparâmetros. Os dados normalizados são então separados em conjuntos de treino e teste com base na proporção previamente especificada (80% e 20% respectivamente) e o modelo é inteiramente recriado com base nos melhores resultados obtidos pela busca dos hiperparâmetros (Figura 22).

Figura 22 - Treinamento do modelo LSTM com os melhores hiperparâmetros

```
# Treinar o modelo final com os melhores hiperparâmetros
best_lr = best_params['lr']
best_window_size = best_params['window_size']

# Pré-processamento dos dados com os melhores hiperparâmetros
scaler = MinMaxScaler(feature_range=(0, 1))
df['preco_norm'] = scaler.fit_transform(df['price'].values.reshape(-1, 1))

# Divisão dos dados em treinamento e teste
proporcao_treinamento = 0.8
n_treinamento = round(proporcao_treinamento * len(df))

dados_treinamento = df.iloc[:n_treinamento]
dados_teste = df.iloc[n_treinamento:]

# Preparação dos dados de treinamento
X_train, Y_train = prepare_data(dados_treinamento['preco_norm'].values, best_window_size)
X_test, Y_test = prepare_data(dados_teste['preco_norm'].values, best_window_size)

# Criação do modelo LSTM com os melhores hiperparâmetros
model = LSTMModel(1, 50, 1)
criterion = nn.MSELoss()
optimizer = Adam(model.parameters(), lr=best_lr)

# Treinamento do modelo com os melhores hiperparâmetros
X_train = torch.Tensor(X_train).unsqueeze(-1)
Y_train = torch.Tensor(Y_train).unsqueeze(-1)
X_test = torch.Tensor(X_test).unsqueeze(-1)
Y_test = torch.Tensor(Y_test).unsqueeze(-1)
```

Fonte: a autora (2023).

O código de treinamento do modelo LSTM termina com o *loop* de épocas e lotes bem como a avaliação do modelo a partir do cálculo da função de perda.

4 APRESENTAÇÃO DOS RESULTADOS

Nessa seção, serão apresentados os resultados obtidos a partir da comparação das métricas da aplicação de ambas as Redes Neurais Artificiais (MLP e LSTM). A primeira etapa foi a estruturação da base de dados que seria utilizada para treinamento e teste dos modelos. Como supracitado, utilizou-se uma base de dados da Kaggle, contendo os preços mensais do *crude oil*, ou petróleo bruto (USD/barril).

Para melhor visualização das informações da base de dados, utilizou-se a biblioteca Pandas para fazer a leitura do arquivo previamente baixado, por meio da função *"read_csv"* é possível transformar os dados em um *DataFrame* para serem utilizados posteriormente em *Python* (Figura 23).

Figura 23 - Importação da base de dados para leitura

```
# Carregar os dados
df = pd.read_csv(r"CRUDE OIL PRICE.csv")
df
```

	date	price	percentChange	change	previous
0	1983-03-30T00:00:00	29.27	NaN	NaN	NaN
1	1983-04-04T00:00:00	30.63	4646.000	1.3600	29.27
2	1983-05-02T00:00:00	30.25	-1241.000	-0.3800	30.63
3	1983-06-01T00:00:00	31.38	3736.000	1.1300	30.25
4	1983-07-01T00:00:00	32.00	1976.000	0.6200	31.38
...	...	...	...	...	...
479	2023-02-01T00:00:00	77.19	-2501.000	-1.9800	79.17
480	2023-03-01T00:00:00	75.80	-1801.000	-1.3900	77.19
481	2023-04-03T00:00:00	76.78	1293.000	0.9800	75.80
482	2023-05-01T00:00:00	68.09	-11318.000	-8.6900	76.78
483	2023-06-01T00:00:00	67.76	-0.479	-0.3263	68.09

484 rows × 5 columns

Fonte: a autora (2023).

Percebeu-se ainda a possibilidade de formatação da primeira coluna de dados (data), já que a informação de tempo em horas não está sendo utilizada. Assim, transformou-se a coluna em questão no formato AAAA-MM-DD (

Figura 24). Ressalta-se ainda que as últimas 3 colunas não foram utilizadas no presente trabalho.

Figura 24 - Formatação da coluna de data

```
df['date'] = pd.to_datetime(df['date'])
df
```

✓ 0.0s

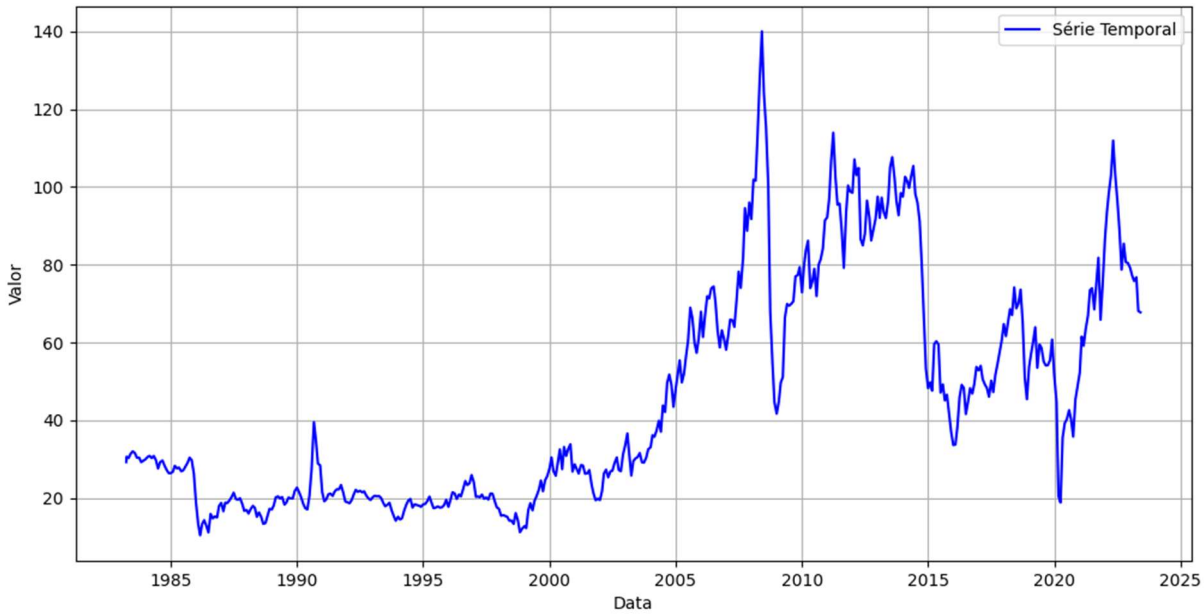
	date	price	percentChange	change	previous
0	1983-03-30	29.27	NaN	NaN	NaN
1	1983-04-04	30.63	4646.000	1.3600	29.27
2	1983-05-02	30.25	-1241.000	-0.3800	30.63
3	1983-06-01	31.38	3736.000	1.1300	30.25
4	1983-07-01	32.00	1976.000	0.6200	31.38
...	...	...	...	...	...
479	2023-02-01	77.19	-2501.000	-1.9800	79.17
480	2023-03-01	75.80	-1801.000	-1.3900	77.19
481	2023-04-03	76.78	1293.000	0.9800	75.80
482	2023-05-01	68.09	-11318.000	-8.6900	76.78
483	2023-06-01	67.76	-0.479	-0.3263	68.09

484 rows × 5 columns

Fonte: a autora (2023).

Para visualização dos dados, utilizou-se a biblioteca *matplotlib* e plotou-se os valores, conforme observado na Figura 25.

Figura 25 - Visualização da série temporal



Fonte: a autora (2023).

4.1 ANÁLISE ESTATÍSTICA DA SÉRIE TEMPORAL

A análise estatística da série temporal dos preços mensais do petróleo se deu apenas para melhor compreensão dos dados, porém vale ressaltar que os resultados dessa análise não foram utilizados nas Redes Neurais Artificiais, pois estas não exigem pressupostos dos dados para sua aplicação.

Inicialmente, usou-se a função “*describe”* para uma análise descritiva das variáveis (Figura 26).

Figura 26 - Aplicação da função *DESCRIBE*

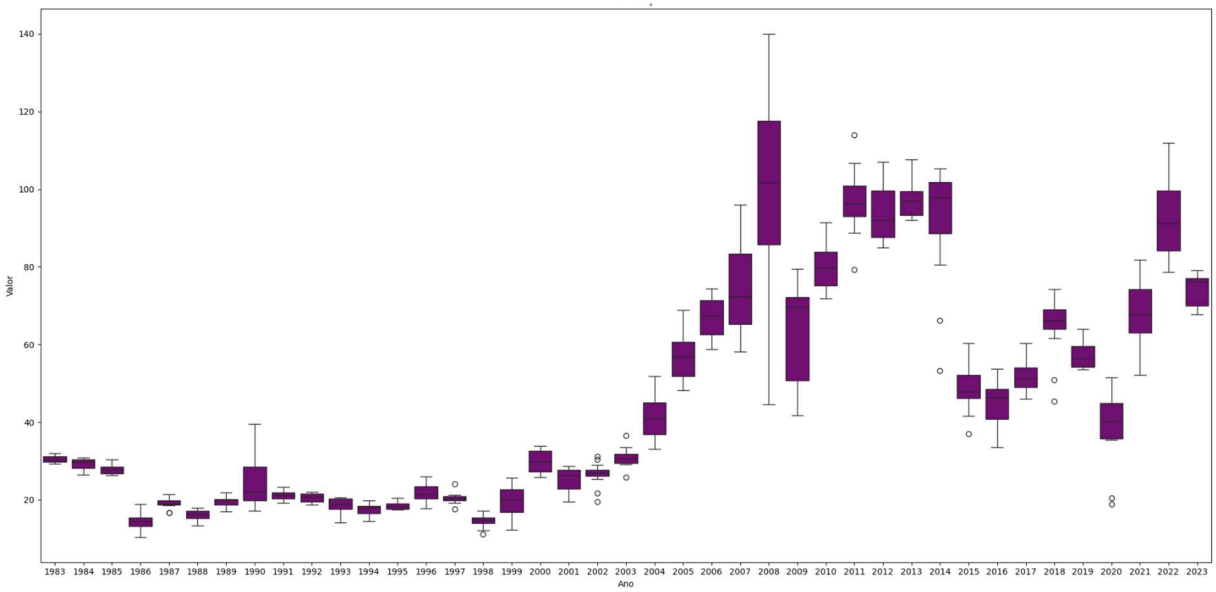
```
df.describe()
✓ 0.0s
```

	date	price
count	484	484.000000
mean	2003-04-17 13:59:00.495867776	45.310517
min	1983-03-30 00:00:00	10.420000
25%	1993-03-24 12:00:00	20.360000
50%	2003-04-16 00:00:00	31.795000
75%	2013-05-09 06:00:00	66.390000
max	2023-06-01 00:00:00	140.000000
std	NaN	28.851433

Fonte: a autora (2023).

Fez-se também uma análise das variações por ano, plotando o gráfico *box plot* para cada período (Figura 27).

Figura 27 - Gráfico de caixa por ano

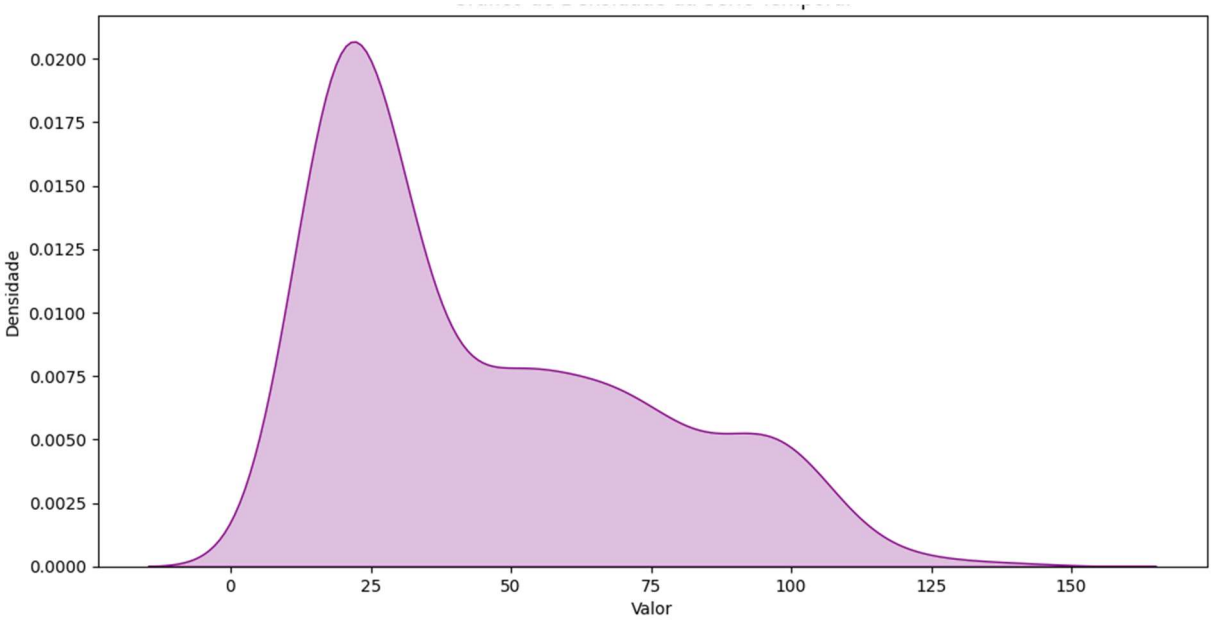


Fonte: a autora (2023).

Pode-se então observar que a variação de valores durante o ano cresceu com o passar do tempo, ou seja, nos anos iniciais da série temporal, havia uma menor variação dos preços. Já nos anos finais da série temporal, a variação se mostrou mais relevante estatisticamente.

Para analisar os valores mais frequentes na série temporal, fez-se um gráfico de densidade (Figura 28), que demonstra a distribuição dos dados.

Figura 28 -Gráfico de densidade da série temporal

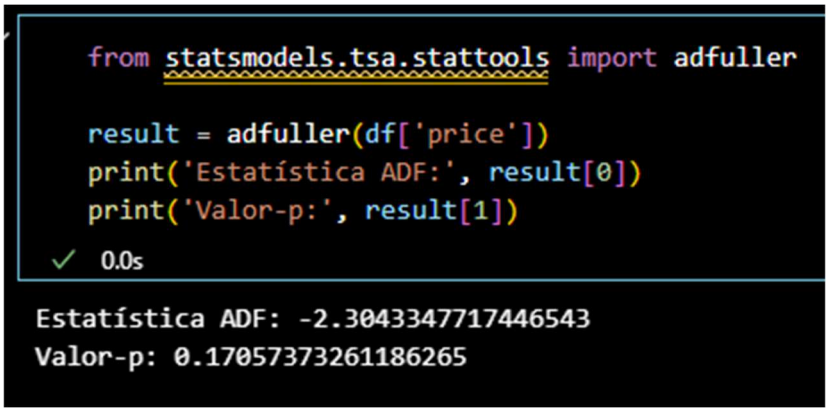


Fonte: a autora (2023).

É possível perceber, então, que poucos dados se encontram nos extremos (0 e 150) e que a faixa de valor com maior representatividade é 15 a 40.

Dando continuidade nos testes estatísticos, analisou-se ainda o Augmented Dickey-Fuller (ADF) – que verifica a estacionariedade de uma série temporal- e o p-valor, que é uma medida estatística que auxilia na tomada de decisão (MUSHTAQ, 2011) (Figura 29). A interpretação desses valores depende do contexto estatístico e do nível de significância escolhido para o teste. O nível de significância é o valor crítico predefinido que determinará quando se rejeitará ou não a hipótese nula no teste estatístico. Para o teste ADF, tem-se como hipótese nula que a série não é estacionária, e considerando um nível de significância de 5% por exemplo, tem-se que p-valor > 0,05, e, portanto, aceitasse da hipótese nula, isto é, a série não é estacionária.

Figura 29 – Aplicação do teste ADF e p -valor



Fonte: a autora (2023).

Fez-se também o teste de Mann-Kendall, que é uma técnica estatística usada para determinar se há tendência monotônica (crescente ou decrescente) em uma série temporal. Esse teste, considera que, na hipótese de estabilidade, a sucessão de valores ocorre de forma independente e a distribuição de probabilidade deve permanecer sempre a mesma (HAMED; HAO, 1998).

O p-valor é calculado a partir da estatística de Mann-Kendall (MK). Ele representa a probabilidade de observar uma estatística de Mann-Kendall extremamente alta (ou baixa) sob a hipótese nula de que não há tendência.

Ao aplicar o teste de MK na série temporal do presente estudo, obteve-se como resultado a observação de uma tendência crescente e com p-valor igual a zero (Figura 30), fato que infere que há evidências estatísticas extremamente fortes de que existe uma tendência nos dados.

Figura 30 - Aplicação do Teste Mann-Kendall e p-valor

```
import pymannkendall

# Aplicar o Teste de Mann-Kendall
result = pymannkendall.original_test(df['price'])

# Interpretar os resultados
if result.trend == "increasing":
    print("Tendência crescente.")
elif result.trend == "decreasing":
    print("Tendência decrescente.")
else:
    print("Tendência indefinida.")

print(f'p-valor: {result.p}')
```

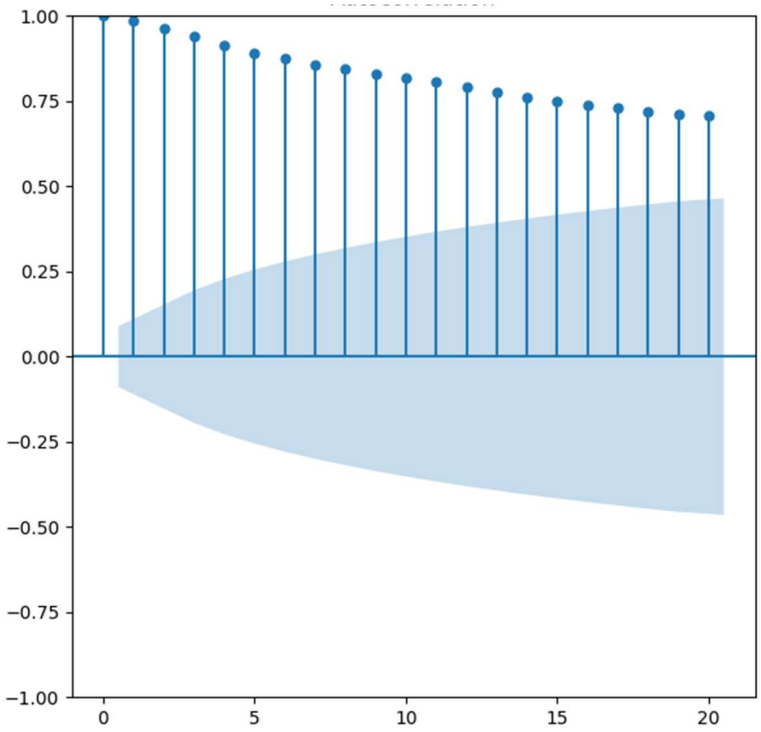
✓ 0.0s

```
Tendência crescente.
p-valor: 0.0
```

Fonte: a autora (2023).

Fez-se também a análise de autocorrelação – que é utilizada para avaliar a correlação entre observações em uma série temporal e as observações em atrasos temporais anteriores - usando a função "*acf*" (Figura 31).

Figura 31 - Análise de autocorrelação

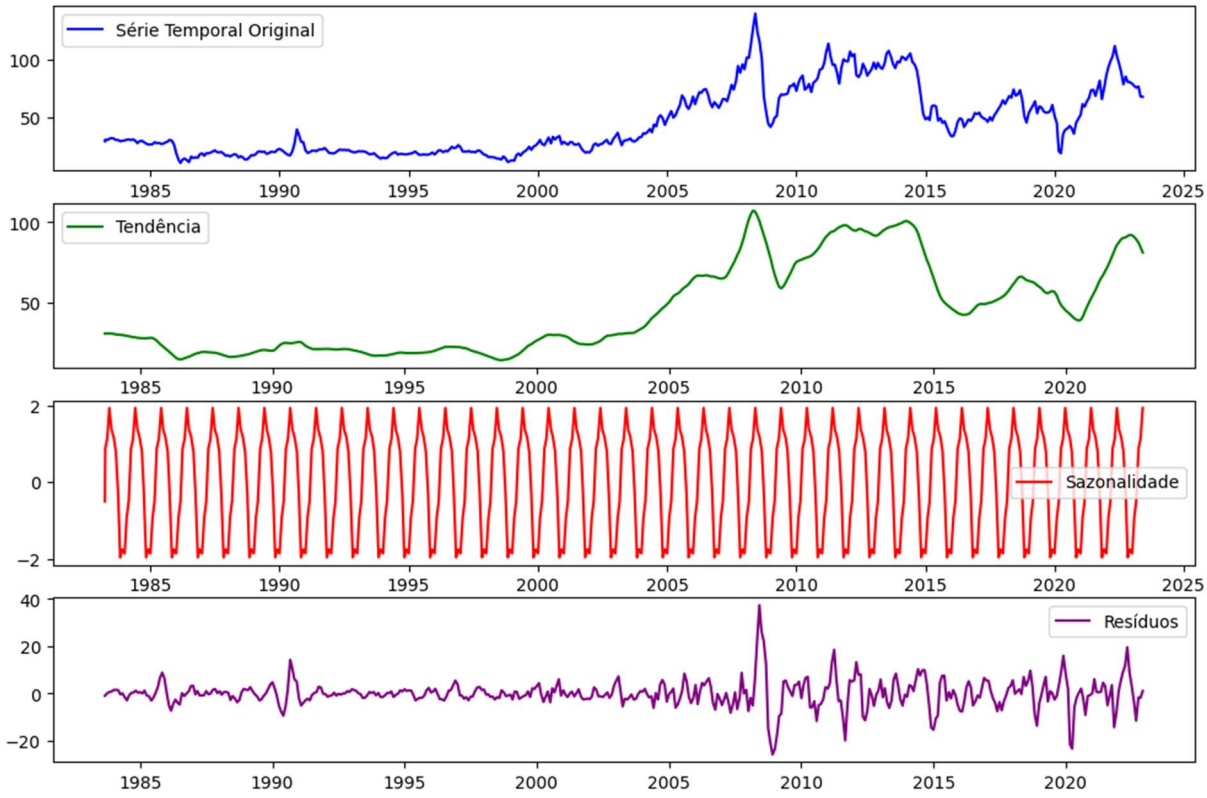


Fonte: a autora (2023).

A partir da interpretação da Figura 31, é possível observar que há explícita correlação positiva dos dados com valores anteriores a ele.

E, por fim, fez-se uma decomposição da série temporal, a fim de facilitar a visualização de importantes atributos dos dados (Figura 32).

Figura 32 - Decomposição da série temporal



Fonte: a autora (2023).

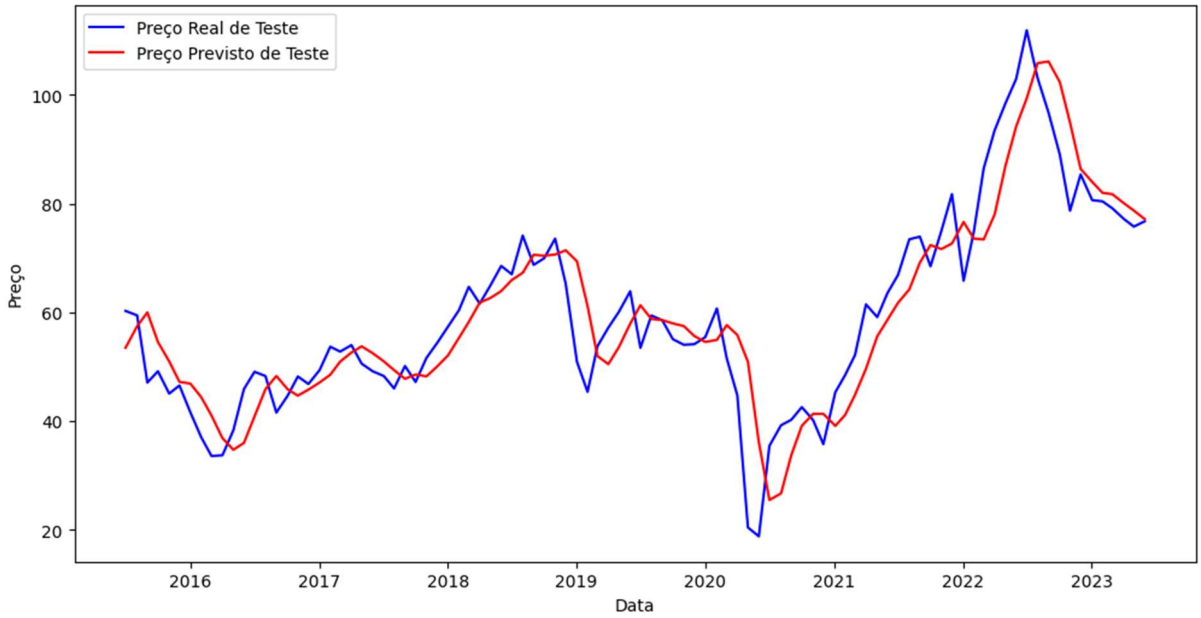
Percebe-se, além da tendência e dos resíduos, a não sazonalidade da série temporal, pelo gráfico é possível notar a variância constante de apenas dois dólares por barril a cada ano, valor irrisório quando comparado ao valor total. Após entendimento completo da base de dados, iniciou-se o processo de desenvolvimento das Redes Neurais Artificiais.

4.2 COMPARAÇÃO DOS RESULTADOS

Na comparação dos resultados obtidos com as redes MLP e LSTM para análise univariada, observa-se que ambas as arquiteturas demonstraram um desempenho

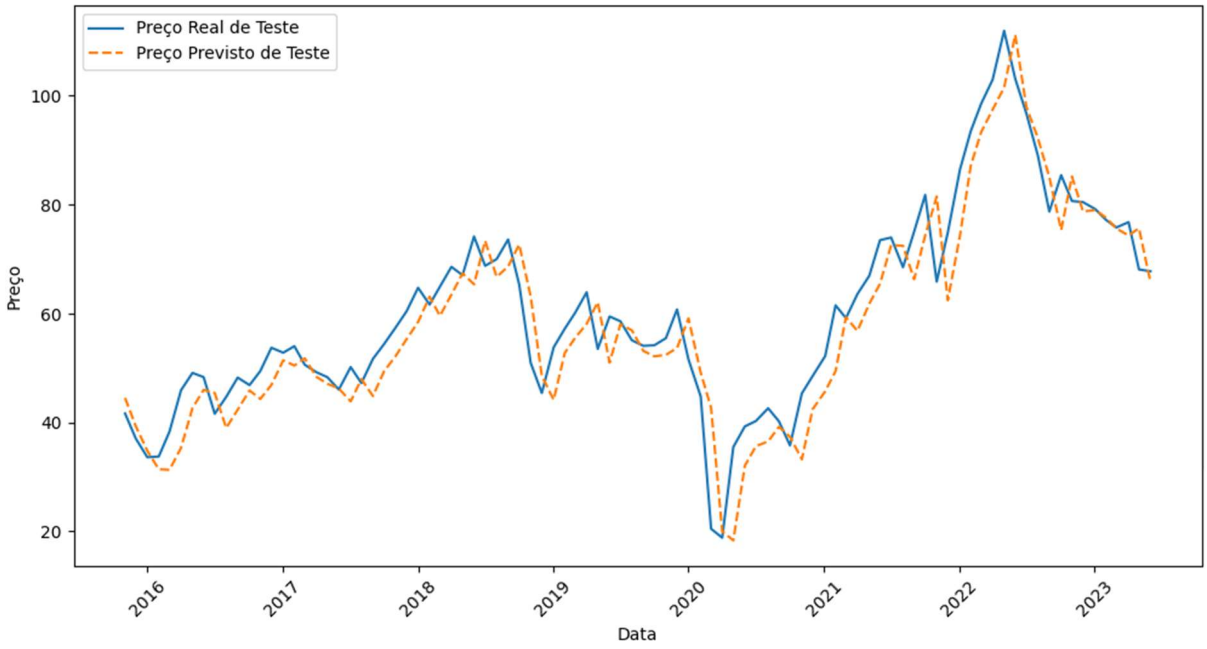
promissor na previsão das variáveis alvo. O MSE obtido em ambas as redes para as variáveis alvo variaram de 36,79 para o modelo MLP e 29,32 para LSTM. Pode-se perceber essa diferença pelos gráficos plotados demonstrados na Figura 33 e Figura 34

Figura 33- Resultados de teste para MLP



Fonte: a autora (2023).

Figura 34 - Resultados de teste para LSTM



Fonte: a autora (2023).

Pode-se ainda perceber que, para ambas as arquiteturas, o melhor tamanho de janela foi 4, ou seja, o melhor número de preços passados usados para prever o próximo é 4. A taxa de aprendizado também se mostrou igual para ambos os modelos, o melhor valor encontrado por ambas as arquiteturas foi 0,01, como mostrado nas tabelas 2 e 3.

Tabela 2 - Melhor combinação de hiperparâmetros para a arquitetura MLP na análise univariada

Tamanho da janela (número de neurônios na camada de entrada)	Taxa de aprendizado	Número de neurônios na primeira camada oculta	Número de neurônios na segunda camada oculta	Melhor MSE
4	0,01	6	2	36,79

Tabela 3 - Melhor combinação de hiperparâmetros para a arquitetura LSTM na análise univariada

Tamanho da janela (número de neurônios na camada de entrada)	Taxa de aprendizado	Número de unidades LSTM	Melhor MSE
4	0,01	50	29,32

4.3 ANÁLISE MULTIVARIADA UTILIZANDO A ARQUITETURA MLP

Para a análise multivariada utilizando a rede MLP, os melhores hiperparâmetros encontrados constam na Tabela 4.

Tabela 4 - Melhor combinação de hiperparâmetros para a arquitetura MLP na análise multivariada

Tamanho da janela (número de neurônios na camada de entrada)	Taxa de aprendizado	Número de neurônios na primeira camada oculta	Número de neurônios na segunda camada oculta	Melhor MSE
3	0,01	1	2	48,63

Fonte: a autora (2023)

Observa-se, então, que, ao adicionar a informação da cotação do dólar em relação ao real, não se obteve uma melhor performance do modelo, ou seja, o dado adicional não ajudou a rede MLP a prever o próximo preço do barril de petróleo.

5 CONSIDERAÇÕES FINAIS

A aplicação de Redes Neurais Artificiais na previsão do preço do petróleo assume um papel de destacada relevância no cenário da análise de mercado e nas decisões de ordem econômica. A antecipação dos valores das *commodities*, a exemplo do petróleo, se caracteriza por uma empreitada complexa, dadas as múltiplas variáveis no contexto, tais como eventos geopolíticos, oscilações na demanda, vicissitudes climáticas, e os efeitos das políticas governamentais, entre outros. Por isso, a utilização de Redes Neurais Artificiais para previsão de seu preço desempenha um papel significativo na análise de mercado e na tomada de decisões econômicas podendo aprimorar a precisão das previsões e proporcionar *insights* valiosos para os tomadores de decisão.

Nesse contexto, as Redes Neurais Artificiais oferecem uma série de prerrogativas que podem elevar a precisão das projeções e proporcionar valiosas perspectivas para os agentes decisórios. O curso desta pesquisa propiciou uma apreciação mais acurada da magnitude e relevância do preço do petróleo, além de um entendimento mais sólido das sutilezas inerentes às séries temporais.

No decorrer da pesquisa, foi ainda possível entender de maneira mais profunda a importância e relevância do preço do petróleo e as características importantes de séries temporais.

À luz dos resultados apresentados no último capítulo, emergem considerações de tamanha importância. Os desdobramentos do estudo revelaram que ambas as Redes Neurais Artificiais exibiram uma proficiência notável ao assimilar e modelar de forma eficaz os elementos do problema em apreço. No entanto, destaca-se que a LSTM, em particular, demonstrou um MSE inferior quando comparada à MLP. Esta disparidade sugere que a LSTM se afigura mais apta a lidar com a natureza sequencial dos dados e a capturar intricadas relações temporais - o que se coaduna de forma notável com a natureza da *commodity* em análise. Ademais, convém realçar o peso da etapa de otimização dos hiperparâmetros no desempenho dos modelos de Redes Neurais Artificiais. A seleção criteriosa dos parâmetros, a exemplo do número de camadas ocultas, o dimensionamento destas camadas e a taxa de aprendizado, teve um impacto direto nos resultados alcançados.

Em última instância, este estudo evidenciou a utilidade das Redes Neurais Artificiais na resolução de problemáticas intrincadas, com a LSTM figurando como uma escolha promissora para lidar com problemas que incorporam séries temporais e relações temporais de maior complexidade. Nesse sentido, salienta-se a imperatividade da escolha minuciosa da arquitetura e dos hiperparâmetros na consecução de resultados satisfatórios. Ressalta-se ainda a relação não existente entre a cotação do dólar em relação ao real e o preço do barril do petróleo, já que essa variável exógena não ajudou a RNA a obter uma melhor performance.

Como próximos passos, é pertinente explorar a perspectiva de desenvolver uma análise multivariada, que leve em consideração outros fatores exógenos, como os preços de outras *commodities*, variáveis geopolíticas, e outros elementos que contribuam para enriquecer a robustez do modelo de previsão.

REFERÊNCIAS

ADDAGATLA, Arun. **Investigating Underfitting and Overfitting**. Geek Culture, 2021. Disponível em: < https://medium.com/geekculture/investigating-underfitting-and-overfitting-70382835e45c>. Acesso em: 20 jun. 2023.

AMBRÓSIO, Paulo. **Redes neurais artificiais no apoio ao diagnóstico diferencial de lesões intersticiais pulmonares**. 2002

BARROS, Evandro Vieira de. **A matriz energética mundial e a competitividade das nações: bases de uma nova geopolítica**. ENGEVISTA, v. 9, n. 1, p. 47-56, junho 2007. Disponível em < https://periodicos.uff.br/engevista/article/view/8802/6270>. Acesso em: 20 out. 2023.

BEN , BALANCO ENERGÉTICO NACIONAL, 2022. Disponível em: < https://www.epe.gov.br/pt/publicacoes-dados-abertos/publicacoes/balanco-energetico-nacional-ben> . Acesso em: 20 jun. 2022.

BEZERRA, Manoel Ivanildo Silvestre; **Apostila de Análise de Séries Temporais**. Curso de Estatística. UNESP., 2006. Disponível em: < https://www.academia.edu/12029946/Apostila_Series_Temporais_UNESP >. Acesso em: 06 jul 2023.

CAVALHEIRO, Everton Angel; CERETTA, Paulo Sergio; TAVARES, Carlos. Eduardo Moreira; TRINDADE, Larissa de Lima. **Previsibilidade de mercados: um estudo comparativo entre Bovespa e s&p500**. Revista Sociais e Humanas, v. 23, n. 1, p. 61-74, 2010. Disponível em: < https://periodicos.ufsm.br/sociaisehumanas/article/view/1255>. Acesso em: 20 jun 2023.

FIORIN, Daniel; MARTINS, Fernando; SCHUCH, Nelson; PEREIRA, Enio. **Aplicações de redes neurais e previsões de disponibilidade de recursos energéticos solares.** Revista Brasileira de Ensino de F´ısica, v. 33, n. 1, 1309 , 2011. Disponível em: < https://www.scielo.br/j/rbef/a/dxMmkDfkLHjWKssfQDbtWfF/?format=pdf&lang=pt>. Acesso em: 20 out. 2023.

FLORES, João. Henrique Ferreira. **Comparação de modelos MLP/RNA e modelos Box-Jenkins em series temporais não lineares**. Dissertação. Universidade Federal do Rio Grande do Sul. Escola de Engenharia. Programa de Pós-Graduação em Engenharia de Produção. 2009. Disponível: < https://lume.ufrgs.br/handle/10183/17150>. Acesso em: 20 jun 2023.

GHARAT, Snehal. **O quê, por que e qual ?? Funções de Ativação**.2019. Disponível em:<https://medium.com/@snaily16/what-why-and-which-activation-functions-b2bf748c0441>. Acesso em: 06 jul.2023

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. **Deep Learning**: 1. Cambridge: MIR press, 2016. Disponível em: <

http://alvarestech.com/temp/deep/Deep%20Learning%20by%20Ian%20Goodfellow,%20Yoshua%20Bengio,%20Aaron%20Courville%20(z-lib.org).pdf>. Acesso em: 20 jun.2023.

GUO, Aiping; JIANG, Ajuan; LIN, J; LI, Xiaoxiao. **Data mining algorithms for bridge health monitoring**. Kohonen clustering and LSTM prediction approaches. J Supercomput n.76, p. 932–947, 2019. Disponível: <https://doi.org/10.1007/s11227-019-03045-8>. Acesso em: 20 jun 2023.

GREFF, Klaus, SRIVASTAVA, Rupesh., KOUTNIK, Jan; STEUNEBRINK, Bas; SCHMIDHUBER, Jurgen. **LSTM: A Search Space Odyssey**. IEEE Transactions on Neural Networks and Learning Systems, 28(10), 2222–2232. 2017. Disponível em: < https://ieeexplore.ieee.org/document/7508408/authors#authors>. Acesso em: 20 DE JUN 2023.

Khaled H. Hamed, A. Ramachandra Rao, **A modified Mann-Kendall trend test for autocorrelated data**, Journal of Hydrology, Volume 204, 1998.

HAYKIN, Simon. **Redes Neurais: princípios e prática**. Tradução de Paulo Martins Engel. 2. ed. Porto Alegre: Bookman, 2001.

HOCHREITER, Sepp; SCHMIDHUBER, Jurgen. **Long short-term memory**. Revista Neural Computation, v. 9 n.8, p. 1735–1780, 1997. Disponível em: < https://direct.mit.edu/neco/article-abstract/9/8/1735/6109/Long-Short-Term-Memory?redirectedFrom=fulltext>. Acesso em: 20 jun. 2023.

IEA, **International Energy Agency**. Disponível em < https://www.iea.org/data-and-statistics/data-tools/energy-statistics-data-browser?country=WORLD&fuel=Energy%20supply&indicator=TESbySource>. Acesso em: 20 jun. 2022.

LACERDA, Daniel Pacheco; DRESCH, Aline; PROENÇA, Adriano; ANTUNES JUNIOR, José Antonio Valle. **Design Science Research: método de pesquisa para a engenharia de produção**. Revista Gest. Prod, v. 20 n. 4, São Carlos, 2013. Disponível em: < https://www.scielo.br/j/gp/a/3CZmL4JJxLmxCv6b3pnQ8pq/>. Acesso em: 20 jun. 2023.

MARTINS, Petrônio Garcia; LAUGENI, Fernando Piero. Administração da produção. 2. ed., rev. aum. e atual. São Paulo: Saraiva, 2005.

MCCULLOCH, Warren; PITTS, Walter**. A Logical calculus of de ideas immanente in nervous activity.** Bulletin of Mathematical Biology. Vol. 52, n. ½, p. 99-115, 1990. Disponível em: < https://www.cs.cmu.edu/~./epxing/Class/10715/reading/McCulloch.and.Pitts.pdf> Acesso em: 20 out. 2023.

MIRANDA, Franciely. Abati; FREITAS, Silvio. Rogério. Correia de; FAGGION, Pedro. Luis. **Integração e interpolação de dados de anomalias ar livre utilizando-se a técnica de RNA e krigagem**. Boletim de Ciências Geodésicas, v.15, n.3, p. 428-

443, 2009. Disponível em: < http://www2.fct.unesp.br/departamentos/cartografia/eventos/2007_II_SBG/artigos/A_011.pdf>. Acesso em: 20 jun 2023.

MORETTIN, Pedro; TOLOI, Clelia. **Análise de Séries Temporais**. São Paulo: Edgard Blücher, 2004.

MUSHTAQ, Rizwan, **Augmented Dickey Fuller Test**, 2011

NUNES, Tercia. Valfridia. Lima. **Método de Previsão de Defeitos em Estradas Vicinais de Terra com Base no Uso das Redes Neurais Artificiais**: Trecho de Aquiraz - CE. 2003. 118 f. Dissertação (Mestrado) - Curso de Engenharia de Transportes, Universidade Federal do Ceará, Fortaleza, 2003. Disponível em: < https://repositorio.ufc.br/handle/riufc/4904>. Acesso em: 20 jun.2023.

OLAH, C; **Understanding LSTM networks**, 2015. Disponível em: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>. Acesso em: 20 jun.2023.

PEDROSA, Oswaldo; CORRÊA, Antônio**. A Crise do Petróleo e os Desafios do Pré-sal**. Boletim de Conjuntura, v. 2, p. 4-14, 2016. Disponível em: < https://repositorio.ufc.br/bitstream/riufc/65559/1/2021_tcc_jcclima.pdf> Acesso em: 20 out. 2023.

RIBEIRO, Athyrson; PAULA, ARAÙJO JUNIOR, Francisco de Paula. **Um estudo comparativo entre cinco métodos de otimização aplicados em uma RNC voltada ao diagnóstico do Glaucoma**. Revista de Sistemas e Computação, Salvador, v. 10, n. 1, p. 122-130, jan./abr. 2020. Disponível em: < http://www.revistas.unifacs.br/index.php/rsc> Acesso em: 20 out. 2023.

SANTOS, Alcione. Miranda dos.; SEIXAS, José Manoel de; PEREIRA, Basília de Bragança.; MEDRONHO, Roberto de Andrade. **Usando redes neurais artificiais e regressão logística na predição da hepatite A**. Revista Brasileira de Epidemiologia, V.8, N.2, P. 117-126, 2005. Disponível em: < https://www.scielo.br/j/rbepid/a/wpHxNfpjJz4k9VHBRrzgVfw/?format=pdf&lang=pt>. Acesso em: 20 Jun. 2023.

WIERINGA, Roel. **Design science as nested problem solving**. Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology, n. 8, p. 1–12 New York: ACM, 2009. Disponível: < https://dl.acm.org/doi/10.1145/1555619.1555630>. Acesso em: 20 jun 2023.

ZHANG, Haotian; ZHANG Lin; JIANG, Yuan. **Overfitting and Underfitting Analysis for Deep Learning Based End-to-end Communication Systems**. 11th International Conference on Wireless Communications and Signal Processing (WCSP), China, p. 1-6, 2019. Disponível em: < https://ieeexplore.ieee.org/abstract/document/8927876>. Acesso em: 20 jun. 2023.