MINISTÉRIO DA EDUCAÇÃO
Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Especialização em Ciência de Dados

Predição de variáveis no processo de branqueamento da celulose por meio de redes neurais artificiais

Paulo Molinar Henrique

João Monlevade, MG
2024

Paulo Molinar Henrique

Predição de variáveis no processo de branqueamento da celulose por meio de redes neurais artificiais

Trabalho de conclusão de curso apresentado ao curso de Ciência de Dados do Instituto de Ciências Exatas e Aplicadas da Universidade Federal de Ouro Preto, como parte dos requisitos necessários para a obtenção do título de Especialista em Ciência de Dados.

Orientador: Profa. Dra. Sarah Negreiros de Carvalho Leite

João Monlevade, MG

2024

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

H519p Henrique, Paulo Molinar.
Predição de variáveis no processo de branqueamento da celulose por meio de redes neurais artificiais. [manuscrito] / Paulo Molinar Henrique. - 2024.
86 f. (Série:)

Orientadora: Profa. Dra. SARAH NEGREIROS DE CARVALHO LEITE.
Produção Científica (Especialização). Universidade Federal de Ouro Preto. Departamento de Engenharia de Produção.
ISBN: .
ISSN: .

1. Celulose - Branqueamento. 2. Celulose - Controle preditivo. 3. Controle de produção. 4. Redes neurais (Computação). 5. Industria de celulose. I. LEITE, SARAH NEGREIROS DE CARVALHO. II. Universidade Federal de Ouro Preto. III. Título.

CDU 658.5:004.8

Bibliotecário(a) Responsável: Flavia Reis - CRB6-2431

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E APLICADAS
DEPARTAMENTO DE ENGENHARIA DE PRODUCAO - ICEA

FOLHA DE APROVAÇÃO

Paulo Molinar Henrique

Predição de Variáveis no Processo de Branqueamento da Celulose Por Meio de Redes Neurais Artificiais

Trabalho de conclusão de curso apresentado ao curso de Especialização em Ciência de Dados da Universidade Federal de Ouro Preto como requisito parcial para obtenção do título de Especialista em Ciência de Dados

Aprovada em 23 de maio de 2024

Membros da banca

Dra. Sarah Negreiros de Carvalho Leite - Orientadora - Instituto Tecnológico de Aeronáutica
Dr. Fernando Bernades de Oliveira - Universidade Federal de Ouro Preto
Dr. Flávio Marcelo Correia - Celulose Nipo Brasileira - CENIBRA

Sarah Negreiros de Carvalho Leite, orientadora do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 29/07/2024

Documento assinado eletronicamente por **Thiago Augusto de Oliveira Silva**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 01/08/2024, às 16:41, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0749227** e o código CRC **7DCB0ACC**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35402-163
Telefone: (31)3808-0817 - www.ufop.br

"Dedico este trabalho à Valéria, Ana Paula e Aline, pilares essenciais da minha família e que me ofereceram incansável apoio. Um reconhecimento especial é dirigido à minha mãe, Mafalda Molinar Henrique, e em memória ao meu estimado pai, Antônio Henrique, cuja integridade e sabedoria continuam a inspirar-me todos os dias."

Agradecimentos

Agradeço profundamente a Deus, Nosso Criador, pelas bênçãos e amor incondicional que permearam toda a minha jornada de vida.

À estimada Profa. Dra. Sarah Negreiros de Carvalho Leite, minha orientadora neste trabalho, expresso minha gratidão pelos preciosos momentos de aprendizado que contribuíram significativamente para o meu crescimento profissional.

À CENIBRA - Celulose NipoBrasileira SA, manifesto minha sincera apreciação por ter concebido e viabilizado a criação da turma de Especialização em Ciência de Dados, bem como por fornecer os recursos necessários para o desenvolvimento deste trabalho.

Aos meus amados pais, Sr. Antônio Henrique e Sra. Mafalda Molinar Henrique, expresso minha profunda gratidão pelo exemplo, amor e dedicação que sempre demonstraram ao longo de toda a minha vida.

À minha esposa Valéria e às minhas queridas filhas, Ana Paula e Aline, dedico meu mais sincero agradecimento pelo amor, apoio e compreensão incondicionais, compreendendo os momentos de ausência decorrentes do empenho nas atividades deste trabalho.

Aos colegas do Departamento de Fabricação, em especial aos engenheiros Leandro Coelho Dalvi, Felipe Paixão Cristelli, Daniel Silva da Costa e Paula Camila Mendes Gomes Gaspar Botrel, manifesto meu reconhecimento pelas valiosas discussões e sugestões que enriqueceram o desenvolvimento deste trabalho.

Aos colegas do Departamento de Manutenção, em especial aos engenheiros Carlos Fernando Soares Fernandes, Flávio Hirotaka Mine e Heitor Andrade Nunes, expresso minha gratidão pelo apoio e pelas discussões técnicas que contribuíram para a condução deste trabalho.

Agradeço também a todos os professores do Curso de Especialização em Ciência de Dados da Universidade Federal de Ouro Preto, cujo conhecimento e orientação foram fundamentais para a realização deste trabalho.

"Em um mundo inundado por dados,
aqueles que sabem destilá-los
detêm o poder de transformar realidades."

Resumo

A unidade de branqueamento é crucial tanto para o custo de produção da celulose quanto para a qualidade do produto final. Diversos fatores influenciam as variações nesse processo, dentre os quais destaca-se a dosagem de reagentes químicos. Essas dosagens são verificadas após o período de reação de cada etapa, que pode durar até duas horas em condições normais de produção. Atrasos na resposta aos ajustes na dosagem dos químicos podem levar a flutuações significativas, afetando diretamente a eficiência do processo de branqueamento, o custo da produção e a qualidade da celulose produzida. Neste estudo, pretende-se modelar redes neurais artificiais do tipo *MultiLayer Perceptron* para predizer o valor de duas variáveis-chaves: *ALVURA* e *NÚMERO KAPPA*, ao final de cada um dos quatro estágios do processo de branqueamento: 1- Estágio de dioxidação a quente (D_{hot}), 2 - Estágio de extração de peróxido de hidrogênio (E_p), 3 - Estágio de dioxidação (D_1) e 4 - Estágio de peroxidação (P). Os resultados demonstram que os modelos propostos foram capazes de gerar predições consistentes, com erro absoluto médio (MAE) e coeficiente de determinação (R^2) satisfatórios em todas as etapas. Desta forma, as predições das variáveis em cada estágio possibilitam a otimização do processo, por meio de estratégias de controle de fluxo dos principais reagentes, diminuindo as flutuações e estabilizando o processo de branqueamento.

Palavras-chaves: Rede Neurais Artificiais, *MultiLayer Perceptron*, Branqueamento da Celulose Kraft, Fabricação de Celulose, Predição de Alvura e Número Kappa em Planta de Branqueamento.

Abstract

The bleaching unit is crucial for both the production cost of pulp and the quality of the final product. Various factors influence variations in this process, among which the dosage of chemical reagents stands out. These dosages are checked after the reaction period of each stage, which can last up to two hours under normal production conditions. Delays in responding to adjustments in the dosage of chemicals can lead to significant fluctuations, directly affecting the efficiency of the bleaching process, production costs, and the quality of the produced pulp. In this study, it is intended to model artificial neural networks of the MultiLayer Perceptron type to predict the value of two key variables: BRIGHTNESS and KAPPA NUMBER, at the end of each of the four stages of the bleaching process: 1 - Hot dioxidation stage (D_{hot}), 2 - Hydrogen peroxide extraction stage (E_p), 3 - Dioxidation stage (D_1), and 4 - Peroxidation stage (P). The results demonstrate that the proposed models were able to generate consistent predictions, with satisfactory mean absolute error (MAE) and coefficient of determination (R^2) in all stages. In this way, the predictions of the variables at each stage enable the optimization of the process, through control flow strategies of the main reagents, reducing fluctuations and stabilizing the bleaching process.

Keywords*: Artificial Neural Network, Multilayer Perceptron, Kraft Pulp Bleaching, Pulp Manufacturing, Prediction of Brigntness and Kappa Number in Bleaching Plant.*

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

$ALVURA$ Alvura da Celulose

D_1 Estágio com Dióxido de Cloro

D_{hot} Estágio com Dióxido de Cloro à quente

E_p Estágio com Hidróxido de Sódio e Peróxido de Hidrogênio

$KAPPA$ NÚMERO KAPPA

P Estágio com Peróxido de Hidrogênio

R^2 Coeficiente de Determinação

ReLu *Rectified Linear Unit*

$TensorFlow$ Software library for machine learning

tanh Tangente Hiperbólica

AdaGrad *Adaptive Gradient Algorithm*

DATALAKE Repositório Centralizado de Dados

DCSs *Distributed Control System*

HEXA Ácidos Hexenurônicos

IA Inteligência Artificial

MAE *Mean Absolute Error*

MLP *Multilayer Perceptron*

MSE *Mean Squared Error*

OPC *Open Platform Communications Unified Architecture*

PIMS *Plant Information Management Systems*

PLCs *Programmable Logic Controller*

RMSProp *Root Mean Square Propagation*

RNA Rede Neural Artificial

SCADA *Supervisory Control And Data Acquisition*

Lista de símbolos

σ	*Desvio padrão das amostras*
C	*Grau Celsius*
ClO_2	*Dióxido de Cloro*
Cs	*Consistência da Polpa de Celulose*
H_2O_2	*Peróxido de Hidrogênio*
H_2SO_4	*Ácido sulfúrico*
kg/cm^2	*Quilograma por centímetro quadrado*
l/min	*Litros por minuto*
m^3/tSA	*Fator de diluição*
μ	*Média das amostras*
NaOH	*Hidróxido de Sódio*
p	*Pressão*
$\%ISO$	*Unidade de alvura*
pH	*Potencial Hidrogeniônico do meio*
$Q1$	*Primeiro quartil*
$Q3$	*Terceiro quartil*
T	*Temperatura*
t/h	*Tonelada por hora*
tSA/h	*Toneladas de celulose seca ao ar por hora*
x	*Variável de entrada*
z	*Valor padronizado da entrada*

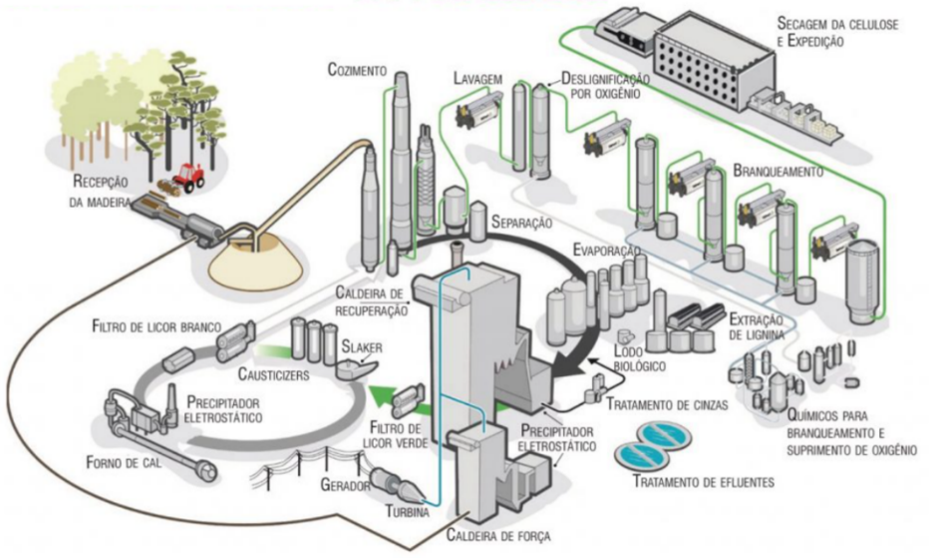
Sumário

1 INTRODUÇÃO

O processo de branqueamento de celulose desempenha um papel crucial na produção de papel e polpa de alta qualidade principalmente a partir da madeira de eucalipto. Durante este processo substâncias químicas são empregadas para remover compostos indesejáveis, sendo a dosagem desses produtos químicos uma variável crítica devido ao seu impacto direto nos custos de produção.

A etapa de branqueamento da polpa pode ter efeitos significativos nas cadeias de celulose, dependendo da sequência de branqueamento e dos oxidantes (químicos) utilizados. Em alguns casos, especialmente em processos realizados a altas temperaturas e em condições alcalinas, há uma propensão à dissolução de hemiceluloses/ácidos urônicos de baixa massa molar, o que penaliza o rendimento do processo. Assim, a busca por processos de branqueamento de baixo impacto ambiental, que apresentem alto rendimento e gerem produtos com propriedades adequadas para usos específicos, é crucial tanto do ponto de vista econômico e de qualidade, quanto da sustentabilidade ambiental. A Figura 1 ilustra a disposição de uma planta de branqueamento no contexto da fábrica de celulose e papel.

Figura 1 – Planta de branqueamento no contexto de uma fábrica de celulose e papel.



Fonte:(BRUMANO, 2023).

O branqueamento se estabelece com o propósito de maximizar a alvura e minimizar a presença de compostos que possam comprometer essa característica, buscando simultaneamente reduzir os custos associados à produção. A escolha das tecnologias de branqueamento é influenciada não apenas pela composição físico-química da matéria-prima, mas também pelas propriedades que se deseja conferir ao papel final, como alvura e resistência. Essa seleção criteriosa ressalta a importância de um equilíbrio entre eficiência produtiva e a obtenção de um produto final de alta qualidade (RAJESH; RAY, 2006).

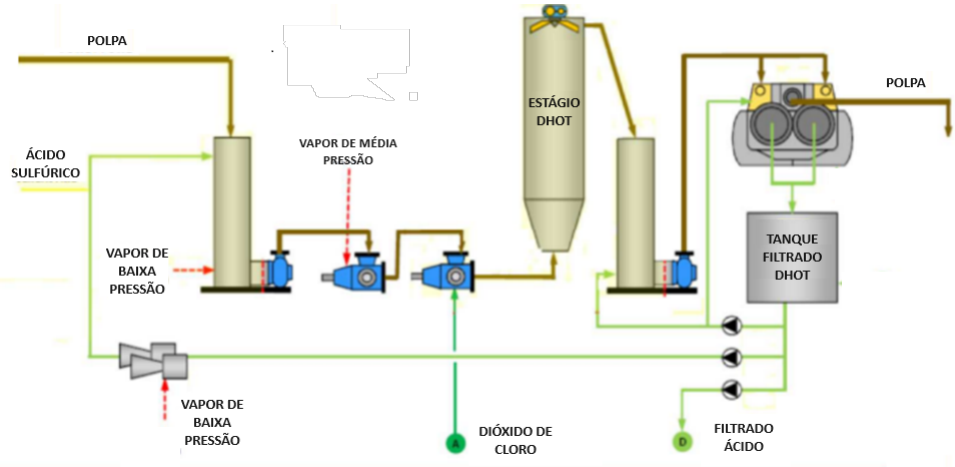
Para uma compreensão mais aprofundada das etapas propostas na metodologia, é essencial entender o funcionamento dos quatro estágios do processo de branqueamento, descritos a seguir.

1.1 ESTÁGIO DE DIOXIDAÇÃO A QUENTE

Na Figura 2 a polpa é alimentada ao *stand pipe* e recebe vapor para controlar a temperatura e ácido sulfúrico (H_2SO_4) para ajuste do *pH* ideal da reação. A bomba que succiona a polpa do *stand pipe* tem a função não apenas de bombeamento, mas também é responsável por aumentar a pressão da polpa na alimentação do estágio. Esses controles, incluindo temperatura , pressão , consistência da polpa e *pH*, são fundamentais para alcançar as condições ideais para a reação de oxidação da lignina presente na polpa e na fibra de celulose.

Após a passagem da massa e dos reagentes pelos misturadores, a polpa é direcionada ao reator para a obtenção do tempo de residência necessário a reação e alcançar os níveis desejados de Alvura da Celulose (*ALVURA*) e NÚMERO KAPPA (*KAPPA*) na saída. Ao término do tempo de reação, a polpa é lavada, prensada para a remoção dos produtos indesejáveis da reação e, em seguida, bombeada para o próximo estágio de branqueamento.

Figura 2 – Fluxograma de processo do estágio D_{hot}.



Fonte: Cenibra.

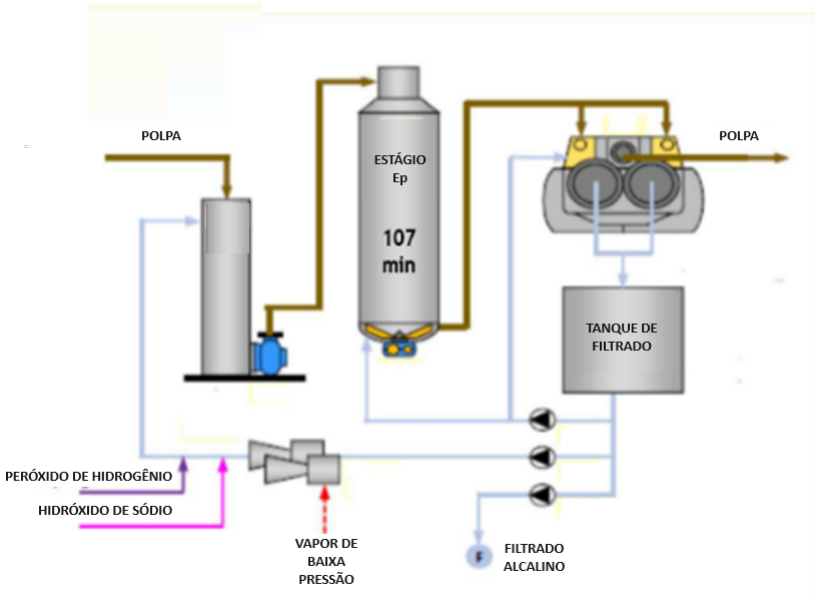
O dióxido de cloro (ClO_2) desempenha um papel fundamental ao oxidar e degradar a lignina, tornando-a solúvel. Os produtos resultantes da degradação dos Ácidos Hexenurônicos (HEXA) no estágio de hidrólise ácida a quente reagem de maneira gradual com o dióxido de cloro (ClO_2), possibilitando sua eliminação na etapa de lavagem após a subsequente acidificação com ácido sulfúrico (H_2SO_4). Neste estágio, as variáveis de processo de maior relevância incluem a *ALVURA* e o NÚMERO *KAPPA* na entrada, além das vazões de produtos químicos (ClO_2 e H_2SO_4), T, p, pH, Cs, e as vazões de vapor de média e baixa pressão.

1.2 ESTÁGIO DE EXTRAÇÃO COM PERÓXIDO DE HIDROGÊNIO

Conforme ilustrado na Figura 3, a polpa é alimentada no *stand pipe*, onde recebe peróxido de hidrogênio, hidróxido de sódio e vapor, desempenhando a função de controlar a temperatura T e ajustar o pH. A bomba responsável por succionar a polpa do *stand pipe*, além de realizar o bombeamento, também eleva a pressão na polpa que alimenta o estágio. Todos esses controles, abrangendo T, p e pH, são cruciais para se alcançar as condições ideais necessárias para que ocorra a reação de extração alcalina com peróxido de hidrogênio.

Após a passagem da massa e dos reagentes pelo rotor da bomba, a polpa é encaminhada ao reator para a solubilização da lignina pela solução alcalina, visando atingir os níveis desejados de *ALVURA* e o NÚMERO *KAPPA* na saída. Ao término do tempo de reação, a polpa passa por processos de lavagem e prensagem para remover os produtos indesejáveis gerados pela reação, sendo, em seguida, bombeada para o próximo estágio de branqueamento.

Figura 3 – Fluxograma de processo do estágio E_p.



Fonte: Cenibra.

A extração alcalina representa um estágio seletivo, caracterizado pela ausência de uma degradação significativa da celulose. Este processo proporciona melhorias notáveis na estabilidade da alvura, opacidade, maciez e algumas propriedades mecânicas da polpa (NAVARRO, 2004; SHACKFORD *et al.*, 2009). Neste estágio, o hidróxido de sódio (NaOH) e o peróxido de hidrogênio (H_2O_2) atuam na solubilização de substâncias que se tornaram solúveis após serem degradadas nos estágios anteriores do processo.

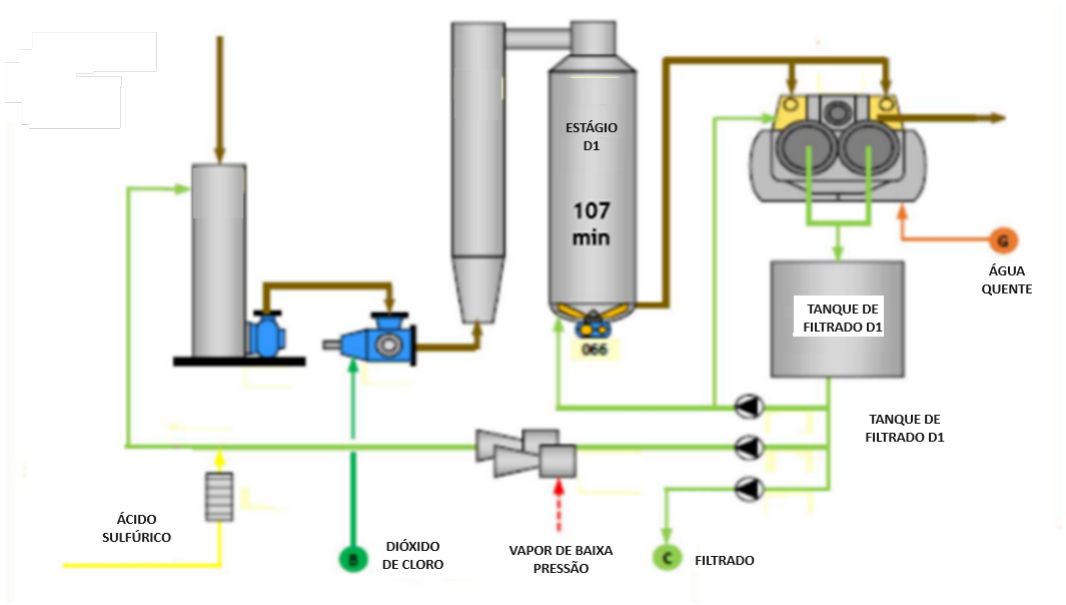
A oxidação da lignina com dióxido de cloro (ClO_2) gera uma fração significativa de lignina de alta massa molecular, a qual se torna solúvel em condições alcalinas. Vale destacar que quanto maior o nível de remoção de lignina durante a extração alcalina, menor será a quantidade de oxidante necessária nos estágios subsequentes. Neste estágio, as variáveis de processo predominantes incluem a $ALVURA$ e o NÚMERO $KAPPA$ na entrada, as vazões de produtos químicos (NaOH e H_2O_2), T, p e pH, a Cs, e as vazões de vapor de baixa pressão.

1.3 ESTÁGIO DE DIOXIDAÇÃO

A Figura 4 mostra a polpa sendo introduzida no *stand pipe*, que recebe vapor para controlar a T e o H_2SO_4 para ajustar o pH. A bomba, além de efetuar o bombeamento, desempenha a função de regular a pressão da polpa. Todos esses controles (T, p e pH) são fundamentais para se alcançarem as condições ideais para a reação de oxidação da lignina residual presente na polpa a na fibra de celulose.

Após a passagem da polpa e dos reagentes pelo misturador, a mistura é encaminhada ao reator para garantir o tempo de residência necessário e obter os níveis desejados de $ALVURA$ e o NÚMERO $KAPPA$ na saída. Ao término do tempo de reação, a polpa é submetida a processos de lavagem e prensagem para remover os produtos indesejáveis gerados pela reação, sendo posteriormente bombeada para o próximo estágio de branqueamento.

Figura 4 – Fluxograma de processo do estágio D_1.



Fonte: Cenibra.

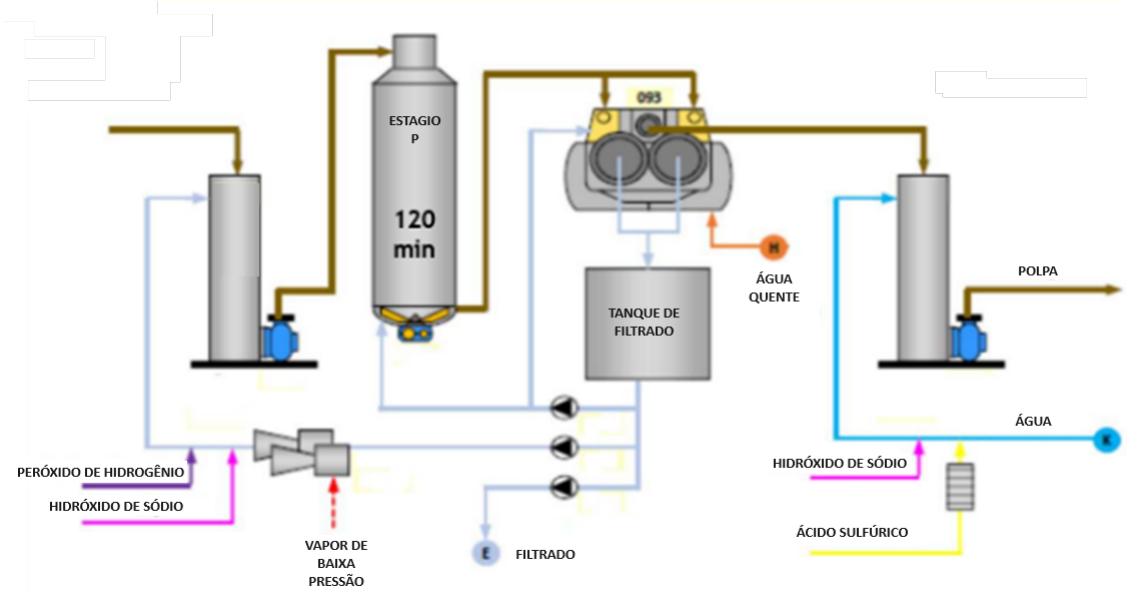
O dióxido de cloro (ClO_2) desempenha um papel crucial ao oxidar e degradar a lignina, tornando-a solúvel. Este gás, absorvido em água gelada e utilizado em pH baixo, é empregado para oxidar e degradar a lignina, tornando-a solúvel. O dióxido de cloro demonstra alta seletividade, promovendo de maneira altamente eficiente a degradação da lignina sem causar degradação significativa dos carboidratos, preservando, assim, a resistência e qualidade da polpa. O Estágio com Dióxido de Cloro (D_1) é recomendado quando o objetivo é produzir polpas com alvuras elevadas e estáveis, limpas e com boas propriedades de resistência. Neste estágio, as variáveis de processo predominantes incluem a *ALVURA* e o NÚMERO *KAPPA* na entrada, as vazões de produtos químicos (ClO_2 e H_2SO_4), T, p e pH, a Cs e as vazões de vapor de média e baixa pressão.

1.4 ESTÁGIO DE PEROXIDAÇÃO

Na Figura 5 a polpa é introduzida no *stand pipe*, que recebe peróxido de hidrogênio (H_2O_2), hidróxido de sódio ($NaOH$) e vapor, desempenhando a função de controlar a temperatura e ajustar o pH. A bomba é responsável por succionar a polpa do *stand pipe* e realizar o bombeamento, tendo também como função aumentar a pressão da polpa na entrada do estágio. Todos esses controles (T, p e pH) são cruciais para se atingir as condições ideais para a reação de peroxidação alcalina com peróxido de hidrogênio.

Após a passagem da massa e dos reagentes pelo rotor da bomba, a polpa é encaminhada ao reator para que ocorra a oxidação da lignina, visando alcançar os níveis desejados de *ALVURA* e o NÚMERO *KAPPA* na saída do estágio. Ao término do tempo de reação, a polpa passa por processos de lavagem e prensagem para os produtos indesejáveis gerados pela reação serem removidos.

Figura 5 – Fluxograma de processo do estágio *P*.



Fonte: Cenibra.

As reações entre o peróxido de hidrogênio e a lignina ainda não foram totalmente esclarecidas, devido à complexidade que envolve os produtos de decomposição do peróxido, incluindo radicais livres ricos em oxigênio (GELLERSTEDT; PRANDA; LINDFORS, 1994).

O peróxido de hidrogênio (H_2O_2) atua deslignificando e modificando a lignina, essencial para o processo de branqueamento da polpa. Em condições alcalinas, ele reage com o íon hidroxila (HO-) formando o ânion perhidroxila (HOO-), um nucleófilo aniônico responsável pelo alvejamento da celulose (SUSS; NIMMERFROH, 1996). Neste estágio, as principais variáveis do processo incluem a *ALVURA* e o NÚMERO *KAPPA* na entrada, as vazões dos produtos químicos NaOH, H_2O_2, a T, p, pH, concentração de sólidos da polpa Cs e as vazões de vapor de baixa pressão.

Após o branqueamento, a celulose é submetida a uma etapa de depuração antes de ser enviada para secagem. Durante esta operação, a água é removida até que a celulose alcance um equilíbrio de 90% de fibras e 10% de água por peso, condições necessárias para a comercialização da celulose.

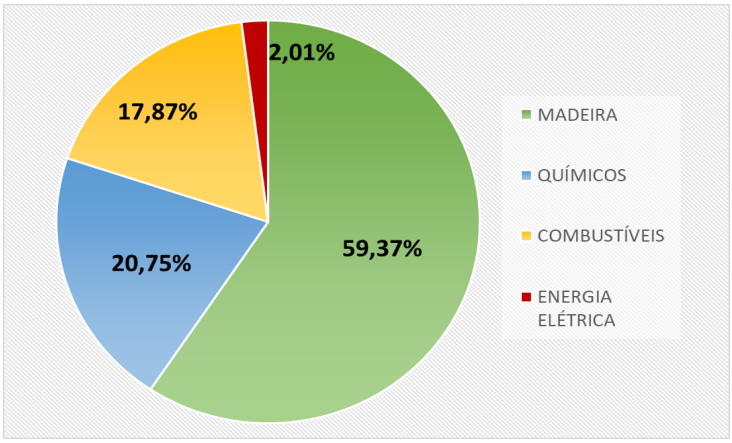
1.5 RELEVÂNCIA DO ESTUDO

A produção de celulose enfrenta diversos desafios que podem ser otimizados e automatizados por meio da Inteligência Artificial (IA) (BAJPAI, 2015).

Este projeto concentra-se na etapa de branqueamento, que se destaca pelo alto custo dos reagentes químicos, um dos principais gastos da indústria de celulose, sendo superado apenas pelo custo da madeira. Portanto, melhorias nesta fase do processo de produção de celulose podem resultar em benefícios significativos.

A Figura 6 mostra a proporção dos custos de uma fábrica de celulose, incluindo os gastos com madeira, combustíveis, reagentes químicos e energia elétrica.

Figura 6 – Comparação dos custos variáveis numa fábrica de celulose.



Fonte: Cenibra.

Os reagentes químicos empregados nas sequências de branqueamento impactam diretamente nas propriedades da polpa ao despolimerizar as cadeias de celulose e, consequentemente, reduzir o comprimento das fibras. Isto resulta em perda de viscosidade e qualidade, além de diminuir o rendimento do branqueamento (COMELATO, 2011).

Dada a importância do setor de celulose para a economia do país, é necessário conduzir pesquisas focadas no aprimoramento do processo produtivo. Os avanços recentes e as novas demandas do mercado intensificaram as expectativas em relação à qualidade da polpa e do papel, destacando a alvura como um dos critérios fundamentais para avaliar as polpas de celulose no cenário comercial.

Diante das contínuas demandas por melhorias operacionais, aumento de desempenho e qualidade da celulose, torna-se essencial desenvolver ferramentas de simulação que permitam testes offline sem comprometer a produção (FERNANDES; CASTRO, 2000).

Do ponto de vista ambiental e da sustentabilidade, este trabalho também se justifica pela racionalização do consumo de produtos químicos e utilidades como água, vapor, energia elétrica e a geração de gases do efeito estufa. Como se propõe a otimização do consumo é esperado um efetivo aumento do rendimento do processo de branqueamento, uma vez que o excesso de reagentes provoca queda na eficiência da planta.

Adicionalmente, este trabalho tem grande importância no entendimento da aplicação das ferramentas de inteligência artificial na aplicação direta ao processo de branqueamento, trazendo novas oportunidades de melhoria de processo e otimização dos recursos e insumos utilizados na planta. Em termos de abrangência, este trabalho proporciona conhecimento suficiente para aplicações em outras áreas e atividades de negócio envolvendo processos industriais.

1.6 REVISÃO DE LITERATURA

No final do século XX, a Rede Neural Artificial (RNA), em particular o modelo *Multi-layer Perceptron* (MLP), se popularizou como ferramenta para solucionar diversas aplicações no contexto de aprendizado de máquina. Inspiradas pelo funcionamento do cérebro humano, essas redes são capazes de aprender padrões complexos nos dados e fazer previsões precisas em uma ampla gama de problemas, abrangendo predição, reconhecimento de padrões, classificação e processamento de dados (CANAKCI; OZSAHIN; VAROL, 2012; TIRYAKI; HAMZAÇEBI, 2014).

Essa técnica evoluiu como um método de modelagem robusto em comparação com abordagens estatísticas ou quantitativas tradicionais (CEYLAN, 2008). Com o emprego de dados representativos e algoritmos ajustados, essas redes são capazes de estabelecer o mapeamento complexo existente entre as variáveis de entrada e de saída (PATTERSON, 1996). De fato, uma das principais vantagens da RNA é sua capacidade de lidar com grandes volumes de dados e extrair informações significativas para tomada de decisão (GARDNER; DORLING, 1998), (ESTEBAN; FERNÁNDEZ; PALACIOS, 2011).

No contexto da indústria de papel e celulose, a MLP pode ser empregada para gerar soluções eficazes em cenários onde prever o comportamento, classificar ou controlar um sistema ou processo se mostra desafiador (HIMMELBLAU, 2000).

De fato, a necessidade de se estudar e de se predizer o comportamento das variáveis por meio de modelagem matemática em fábricas de celulose tem sido o motivo de inúmeras pesquisas ao longo dos anos (HARKONEN, 1987); (MICHELSEN; FOSS, 1996); (QIAN *et al.*, 1997); (AL-AWAMI; SIDRAK, 1998); (WISNEWSKI; III, 1998); (III; KAYIHAN, 1999); (FERREIRA, 2000); (WISNEWSKI; DOYLE, 2001); (AGUIAR; FILHO, 2001); (KAYIHAN, 2002); (CARDOSO *et al.*, 2002); (POLIT; ESTABEN; LABAT, 2002); (QUEIROZ *et al.*, 2004); (DUFOUR *et al.*, 2005); (POUGATCH; SALCUDEAN; GARTSHORE, 2006); e (PADHIYAR *et al.*, 2006).

Em particular, no estudo de RIBEIRO (2007) foi realizada a modelagem do sistema de pré-branqueamento, utilizando redes neurais artificiais para prever o NÚMERO *KAPPA*. Além das variáveis do processo, foram incorporadas análises dos espectros de frequência da absorbância do infravermelho nos cavacos de madeira. O modelo resultante demonstrou uma notável capacidade de predição do NÚMERO *KAPPA*, reduzindo significativamente sua variabilidade. Os resultados da previsão alcançados foram altamente satisfatórios, apresentando características explicativas robustas e validando o modelo. O estudo de (RIBEIRO, 2007) sugere que a abordagem com redes neurais artificiais pode ser eficaz para modelar os efeitos das espécies de madeira, índice de tração e alvura na produção de celulose. Oferecendo uma base para tomadas de decisão preliminares, especialmente em contextos onde o índice de tração e as propriedades de brilho do papel são críticas. Além disso, os modelos propostos proporcionaram economia de tempo, redução no consumo de materiais experimentais e diminuição dos custos de projeto.

Outro resultado interessante foi relatado por (ZAINUDDIN *et al.*, 2011). Os autores conseguiram prever os valores do índice de tração com uma precisão de 96,47% no processo de polpação de folhas de dendezeiro utilizando um modelo de RNA, exemplificando o potencial dessas tecnologias na otimização do branqueamento de polpas.

Diversos parâmetros do processo de branqueamento impactam significativamente o NÚMERO *KAPPA* e a *ALVURA* da polpa branqueada. A análise detalhada do efeito individual de cada parâmetro sobre essas características revela-se uma tarefa onerosa, exaustiva e de longa duração. Diante dessa realidade, a adoção de abordagens mais custo-efetivas para se alcançarem os resultados almejados torna-se crucial para superar tais desafios. Contudo, as informações disponíveis sobre a aplicabilidade de modelos de RNA para prever o impacto de diferentes parâmetros do processo no índice de tração e na alvura das polpas branqueadas ainda são limitadas na literatura.

1.7 OBJETIVOS

Este trabalho propõe o desenvolvimento de um modelo preditivo das variáveis *ALVURA* e o NÚMERO *KAPPA* de cada um dos quatro estágios da planta de branqueamento da CENIBRA - Celulose NipoBrasileira SA., empregando Redes Neurais Artificiais do tipo MLP. Todo o desenvolvimento e ajuste dos hiperparâmetros dos modelos empregou dados reais coletados na planta de branqueamento. Pretende-se que as predições permitam o ajuste das malhas de controle de vazão de reagentes, otimizando o controle e estabilizando a produção, além de melhorar a qualidade da polpa produzida e reduzir os custos de produção.

Para cumprimento do objetivo geral é necessário atender aos seguintes objetivos específicos:

- Levantamento bibliográfico abrangente sobre os conceitos e técnicas das redes neurais MLP, explorando sua aplicação em processos industriais.
- Identificação da influência de cada variável monitorada durante os estágios de branqueamento da celulose sobre as predições das variáveis *ALVURA* e o NÚMERO *KAPPA*.
- Desenvolvimento e treinamento de redes neurais MLP utilizando uma base de dados real, com foco na otimização dos parâmetros do modelo para alcançar o melhor desempenho. A avaliação foi realizada utilizando métricas de qualidade, como erro absoluto médio e Coeficiente de Determinação (R^2).
- Avaliação da adequação e generalização dos modelos propostos, por meio da comparação entre os valores preditos e os valores reais;
- Avaliação da viabilidade da aplicação dos modelos de predição propostos para a predição da *ALVURA* e do valor do NÚMERO *KAPPA* em cada uma das quatro etapas do processo de branqueamento, visando a otimização da produção.

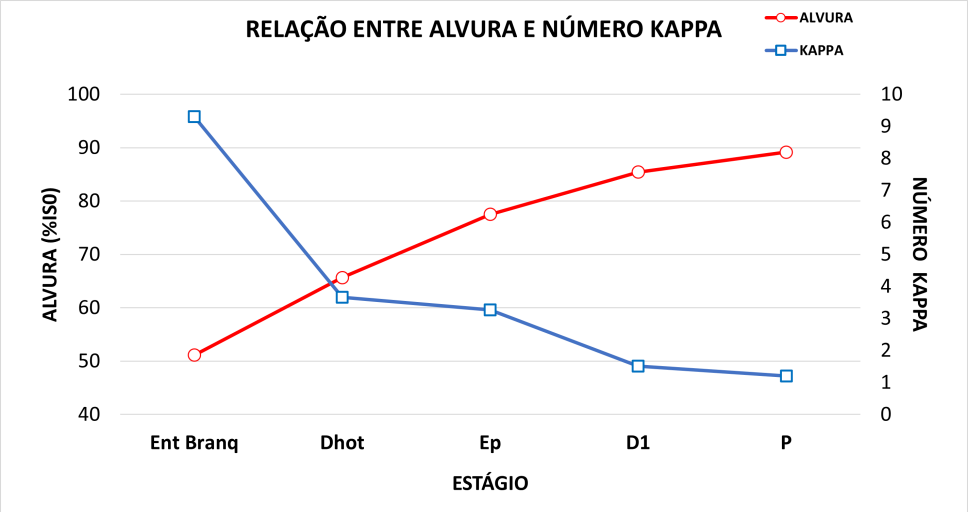
1.8 ORGANIZAÇÃO DO TRABALHO

Para melhor clareza, esta monografia foi organizada em 4 capítulos. O Capítulo 1 apresenta a introdução com a motivação, uma breve revisão de literatura e objetivos do projeto. No Capítulo 2 é apresentada a metodologia empregada no desenvolvimento do modelo preditivo. O Capítulo 3 traz os resultados e discussões e o Capítulo 4 apresenta as conclusões finais e trabalhos futuros.

2 METODOLOGIA

A *ALVURA* é um indicador de qualidade da celulose, aumentando progressivamente à medida que a polpa passa por cada estágio do processo de branqueamento. Em contraste, o NÚMERO *KAPPA*, que pode conferir cor indesejada à celulose, é reduzido em cada estágio devido à ação de agentes oxidantes. A Figura 7 apresenta a relação de proporcionalidade inversa existente entre a *ALVURA* e o NÚMERO *KAPPA*.

Figura 7 – Comportamento das variáveis *ALVURA* e *NÚMERO KAPPA* ao longo dos estágios do processo de branqueamento.

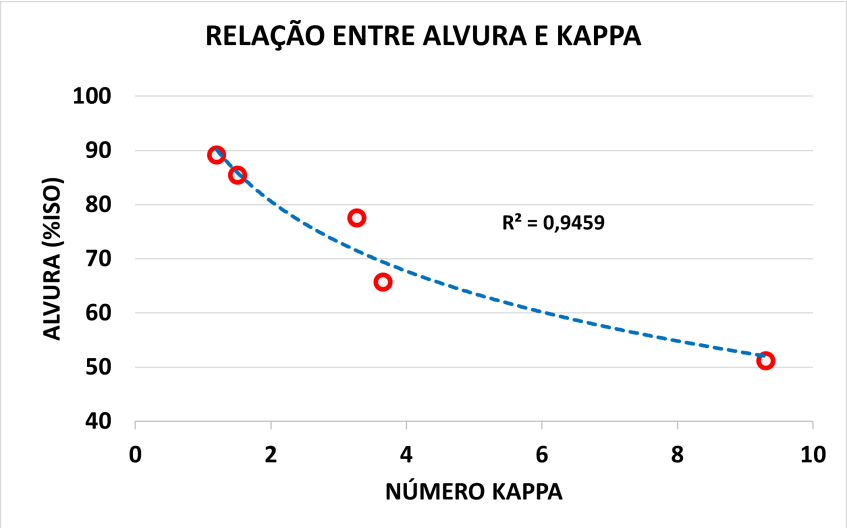


Fonte: Cenibra.

A correlação entre a *ALVURA* e o NÚMERO *KAPPA* pode ser visualizada na Figura 8. O gráfico mostra uma curva decrescente, indicando que conforme o NÚMERO *KAPPA* aumenta, a *ALVURA* diminui. O R^2 de 0,95 indica uma alta relação entre o NÚMERO *KAPPA* e a *ALVURA*, indicando que a variação na *ALVURA* pode ser explicada pela variação do NÚMERO *KAPPA*.

Os pontos vermelhos no gráfico indicam as médias nos respectivos estágios, que se ajustam bem à curva de tendência, reforçando a correlação inversa entre os dois parâmetros. Em resumo, o gráfico mostra uma forte correlação negativa entre o NÚMERO *KAPPA* e a *ALVURA* da polpa de celulose. Este tipo de análise é crucial para a indústria papeleira, pois permite ajustar processos de branqueamento para otimizar a qualidade do produto final.

Figura 8 – Correlação entre a *ALVURA* e o NÚMERO *KAPPA* ao longo dos estágios de branqueamento.



Fonte: Cenibra.

A Figura 9 mostra a evolução da *ALVURA* da polpa ao longo dos diferentes estágios de processamento. No cozimento, a polpa de eucalipto apresenta uma coloração marrom escura, indicando a presença significativa de lignina e outros componentes não celulósicos. No pré-branqueamento, a polpa passa por um processo inicial de branqueamento, resultando em uma coloração marrom mais clara em comparação ao estágio de cozimento, sugerindo uma redução no conteúdo de lignina. No Estágio com Dióxido de Cloro à quente (D_{hot}), a polpa torna-se consideravelmente mais clara, apresentando uma tonalidade bege claro, o que indica um avanço significativo na remoção de lignina. No Estágio com Hidróxido de Sódio e Peróxido de Hidrogênio (E_p), a polpa se apresenta ainda mais clara, próxima ao branco, indicando que a maior parte da lignina foi removida e que a celulose está se tornando mais pura. No estágio D_1, a polpa continua a clarear, apresentando uma alvura quase completa. Nessa fase, a polpa é quase totalmente branca.

Figura 9 – Foto da polpa de celulose evidenciando a evolução da *ALVURA* ao longo dos estágios do branqueamento.

Fonte: Cenibra.

Finalmente, no Estágio com Peróxido de Hidrogênio (*P*), a polpa atinge a alvura de mercado, aparecendo completamente branca, o que indica que está no estágio final de branqueamento, com a remoção máxima de impurezas. Em resumo, a Figura ilustra claramente o aumento progressivo da alvura da polpa através dos diferentes estágios de processamento, desde o material bruto até a polpa totalmente branqueada.

2.1 MÉTRICAS DE ERROS

Diversas métricas permitem quantificar a diferença entre os valores da saída '*REAL*' e '*PREVISTO*', indicando a adequação do modelo. Dentre elas, pode-se citar o R^2, *Mean Absolute Error* (MAE) e *Mean Squared Error* (MSE), calculadas conforme as expressões:

$$R^2 = \frac{\sum_{i=1}^{n} (y_i - \hat{y})^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \tag{2.1}$$

$$MAE = \frac{1}{n} \cdot \sum_{i=1}^{n} |(y_i - \bar{y})| \tag{2.2}$$

$$MSE = \frac{1}{n} \cdot \sum_{i=1}^{n} (y_i - \bar{y})^2 \tag{2.3}$$

Sendo que n é o número de instâncias, y_i é o i-ésimo valor real, $\hat{y}$ é a i-ésima predição e $\bar{y}$ é o valor médio de y.

O R^2 é uma medida útil para avaliar a qualidade do ajuste de um modelo de regressão aos dados observados. Quanto mais próximo de 1, melhor o modelo explica a variação nos dados dependentes. Em outras palavras, ele indica a proporção da variabilidade dos dados dependentes que é explicada pelo modelo de regressão. Portanto um R^2 de 0,80, significa que 80% da variação nos dados dependentes é explicada pelo modelo de regressão, enquanto que os outros 20% são devidos a outras variáveis ou a erros aleatórios.

Já o MAE estima a média das diferenças absolutas entre as predições do modelo e os valores reais. Quanto menor o valor do MAE, melhor é a performance do modelo indicando que ele está fazendo previsões mais próximas dos valores reais. É uma métrica robusta e fácil de ser interpretada, pois está na mesma unidade que os dados originais, o que facilita a compreensão do erro do modelo.

O MSE indica o quanto as predições de um modelo se desviam dos valores verdadeiros ao quadrado, fornecendo uma medida do erro médio quadrático das previsões. Quanto menor o valor do MSE, mais próximo as previsões do modelo estão dos valores reais, indicando uma melhor precisão do modelo.

2.2 BASE DE DADOS

Neste trabalho foram empregados os dados do processo de branqueamento provenientes da Planta de Branqueamento 2 da CENIBRA - Celulose NipoBasileira SA. A empresa fica localizada na cidade de Belo Oriente, leste do estado de Minas Gerais e iniciou suas operações em 1973, Atualmente, a empresa possui duas linhas de produção, sendo cada uma equipada com quatro estágios de branqueamento. A capacidade produtiva total da CENIBRA é de 1,23 milhão de toneladas de celulose por ano de fibra de eucalipto.

Na fase inicial de extração de dados, o objetivo é preparar-los para análises mais aprofundadas e identificar padrões nas etapas seguintes. Isso envolve acessar e extrair informações de um repositório de dados cronológicos conhecido como Repositório Centralizado de Dados (DATALAKE), associado à fábrica em questão, por meio do *Plant Information Management Systems* (PIMS).

Foram utilizados dados brutos das médias horárias das variáveis de processo na entrada e saída dos quatro estágios da planta de branqueamento. Esses dados foram obtidos diretamente do sistema *PIMS*, que é um sistema de aquisição de dados online, responsável por receber dados do processo por meio de instrumentos de medição instalados na planta, armazená-los em um banco de dados único e disponibilizá-los por meio de diversas ferramentas. A partir de uma estação de trabalho, pode-se visualizar tanto os dados em tempo real como históricos, podendo montar tabelas, gráficos de tendência, telas sinópticas e relatórios dinâmicos, concentrando a informação e possibilitando uma visão unificada de todo o processo produtivo (CARVALHO *et al.*, 2013).

O PIMS é capaz de concentrar os dados de diferentes fontes, transformando-os em informação e conhecimento. Os dados são fornecidos por instrumentos instalados em linha, coletados em uma frequência pré-definida (uma leitura por minuto) para cada instrumento de medição, por meio de diversas fontes como *Programmable Logic Controller* (PLCs), *Supervisory Control And Data Acquisition* (SCADA), *Distributed Control System* (DCSs) e geralmente a partir da interface *Open Platform Communications Unified Architecture* (OPC). Os dados foram baixados para uma planilha em *Excel* para alimentar o código fonte desenvolvido na linguagem Python.

Os dados brutos coletados são médias horárias, abrangendo o período de 01/01/2021 a 30/09/2023. A seleção desse intervalo foi estrategicamente baseada na sazonalidade da matéria-prima (qualidade da madeira fornecida à fábrica), nas condições climáticas e na inclusão de todas as horas do dia. Esta abordagem garante que os dados sejam representativos e não estejam influenciados por períodos atípicos, promovendo a causalidade, a generalização e a replicabilidade das informações coletadas.

Considerando a importância do processo de branqueamento, todas as variáveis de entrada de cada estágio, medidas por instrumentos *online*, foram consideradas. Estas variáveis incluem o ritmo de produção, as vazões de produtos químicos e vapor, a Temperatura (T) , o pH, a Pressão (p) e a Consistência da polpa (Cs), representando o processo e influenciando as variáveis de saída (variáveis dependentes), tais como a $ALVURA$ e o NÚMERO $KAPPA$ sendo estas o foco principal deste estudo.

As Tabelas 2 a 5 indicam o conjunto de variáveis monitoradas em cada estágio específico do processo de branqueamento envolvendo a produção de celulose, divididas em variáveis de entrada 'x' e saída 'y'.

Tabela 2 – Variáveis de processo do estágio D_{hot}.

	ESTÁGIO D_{hot}				
	Variável	**TAG**	**Unidade**	**Valor Mínimo**	**Valor Máximo**
x	Produção Entrada	221FX04A	tSA/h	85	94
	Fator de diluição da prensa lavadora	218FX02FD	m^3/tSA	1,5	3,5
	Temperatura do estágio	221TC93	C	80	100
	Pressão do estágio	221PI33	kg/cm^2	0	10
	Consistência da polpa de celulose	221AC45	%	11	15
	pH da polpa de celulose	221AC03	–	3	4
	Vazão de vapor de baixa pressão	218FX80	t/h	0	30
	Vazão de vapor de média pressão	221FC34	t/h	0	20
	Vazão de ácido sulfúrico	221FC35	l/min	0	10
	Vazão de dióxido de cloro	221FX32 CPV2	t/h	0	30
	Alvura da celulose na entrada do estágio	221AI60A_A	$\%ISO$	40	
	N° kappa da celulose na entrada do estágio	221AI60A_K	–	0	15
y	Alvura da celulose na saída do estágio	221AI60B_A	$\%ISO$	55	80
	N° kappa da celulose na saída do estágio	221AI60B_K	–	0	5

Fonte: Cenibra.

Tabela 3 – Variáveis de processo do estágio E_p.

	ESTÁGIO E_p				
	Variável	TAG	Unidade	Valor Mínimo	Valor Máximo
x	Produção Entrada	221FX04A	tSA/h	85	94
	Temperatura do estágio	221TC94	C	75	150
	Pressão do estágio	221PI50	kg/cm^2	5	
	Consistência da polpa de celulose	221AC44	%	8	15
	pH da polpa de celulose	221AC18	–	8	
	Vazão de peróxido de hidrogênio	221FX88 CPV1	t/h	2	10
	Vazão de hidróxido de sódio	221FX47 CPV1	t/h	5	
	Alvura da celulose na entrada do estágio	221AI60B_A	$\%ISO$	50	
	N° kappa da celulose e na entrada do estágio	221AI60B_K	–	0	
y	Alvura da celulose na saída do estágio	221AI60C_A	$\%ISO$	60	
	N° kappa da celulose e na saída do estágio	221AI60C_K	–	0	

Fonte: Cenibra.

Tabela 4 – Variáveis de processo do estágio D_1.

	ESTÁGIO D_1				
	Variável	TAG	Unidade	Valor Mínimo	Valor Máximo
x	Produção Entrada	221FX04A	tSA/h	85	94
	Temperatura do estágio	221TC08	C	80	90
	Pressão do estágio	221PI48	kg/cm^2	5	
	Consistência da polpa de celulose	221AC63	%	10	15
	pH da polpa de celulose	221AC15	–	2	14
	Vazão de ácido sulfúrico	221FC81	l/min	0	10
	Vazão de dióxido de cloro	221FX26D CPV1	t/h	0	15
	Alvura da celulose na entrada do estágio	221AI60C_A	$\%ISO$	70	
	N° kappa da celulose na entrada do estágio	221AI60C_K	–	3	
y	Alvura da celulose na saída do estágio	221AI60D_A	$\%ISO$	80	
	N° kappa da celulose na saída do estágio	221AI60D_K	–	0	

Fonte: Cenibra.

Tabela 5 – Variáveis de processo do estágio P.

	ESTÁGIO P				
	Variável	TAG	Unidade	Valor Mínimo	Valor Máximo
x	Produção Entrada	221FX04A	tSA/h	85	94
	Temperatura do estágio	221TI38	C	70	90
	Pressão do estágio	221PI55	kg/cm^2	0	
	Consistência da polpa de celulose	221AC67	%	5	15
	pH da polpa de celulose	221AC19	–	9	
	Vazão de vapor de média pressão	221FC54	t/h	0	30
	Vazão de Peróxido de hidrogênio	221FX87 CPV2	t/h	2	7
	Vazão de Hidróxido de sódio	221FX120 CPV2	t/h	2	7
	Alvura da celulose na entrada do estágio	221AI60D_A	$\%ISO$	80	
	N° kappa da celulose ena entrada do estágio	221AI60D_K	–	0	3
y	Alvura da celulose na saída do estágio	221AI60E_A	$\%ISO$	85	
	N° kappa da celulose ena saída do estágio	A221I60E_K	–	0	3

Fonte: Cenibra.

Neste projeto, os códigos foram desenvolvidos na linguagem Python versão 3.4, utilizando o ambiente do *Google Colab* para implementar as redes neurais do tipo MLP.

2.3 PRÉ-PROCESSAMENTO

Os dados utilizados neste projeto foram provenientes de medições diretas realizadas por instrumentos instalados em linha na planta. Dada a natureza dos dados brutos, que são propensos à falhas, é imperativo realizar um tratamento adequado. Durante os períodos de início e parada, programados ou não, da planta de branqueamento, a operação é instável resultando em variações que não representam o processo como um todo. Para minimizar esse problema, optou-se por excluir as informações desses períodos, garantindo a mensurabilidade dos dados.

Para isto, foi gerada uma matriz para cada um dos quatro estágios do branqueamento contendo todos os dados de entrada e saída. Os dados brutos foram carregados no Python por meio de uma planilha em *Excel* (Figura 44) no Anexo B.

Na sequência, realizou-se uma análise dos dados para identificar e remover *outliers*, isto é, dados que não seguem o padrão esperado da planta. Essa etapa incluiu a utilização de histogramas para auxiliar na detecção dos *outliers*. Desta forma, eliminou-se os dados coletados que, por algum erro de leitura ou de condições, pudessem não corresponder a condição real da planta.

Visando garantir a confiabilidade dos dados de processo, foi estabelecido um limite mínimo de produção de 85 tSA/h (toneladas de celulose seca ao ar por hora). Todos os dados relativos à produção abaixo desse limite, bem como valores negativos e faltantes, foram eliminados. Ao manter um ritmo de produção acima de 85 tSA/h na planta de branqueamento, pode-se assegurar com relativa precisão que todas as variáveis estão estabilizadas e o processo sob controle. Os demais filtros para a exclusão de dados estabelecidos em cada estágio do branqueamento são apresentados nas Figuras 40, 41, 42 e 43 apresentadas no Anexo A.

A Tabela 6 apresenta o quantitativo de amostras antes e após a filtragem para os diferentes estágios D_{hot}, E_p, D_1 e P. Observa-se que apesar da redução da quantidade de amostras com o emprego dos filtros, ainda é possível garantir uma ampla representatividade do comportamento das variáveis em cada estágio.

Tabela 6 – Comparação da quantidade de amostras antes e após a utilização dos filtros.

Estágio	Número de amostras iniciais	Número de amostras após a filtragem
D_{hot}	23.300	8.262
E_p	23.300	12.218
D_1	23.300	8.093
P	23.300	6.502

Fonte: Próprio autor.

2.3.1 PADRONIZAÇÃO DOS DADOS

Como as diferentes variáveis apresentam diferentes ordens de grandeza, é imperativo realizar uma padronização das amostras, de modo a permitir que o modelo de predição possa ser treinado de maneira mais coerente. Neste trabalho, foi empregada a função *StandardScaler* da biblioteca *Sklearn*. Nesse processo, cada amostra é calculada como:

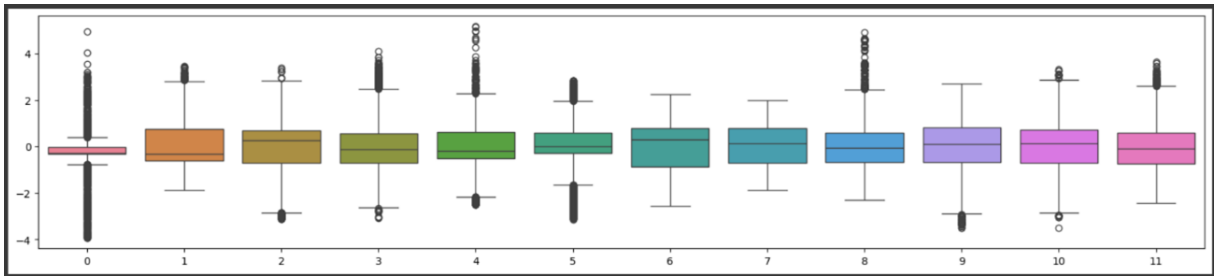
$$z = (x - \mu)/\sigma \tag{2.4}$$

sendo x a variável de entrada, μ a média das amostras, σ o desvio padrão e z o valor padronizado da entrada. Essa abordagem assegura uma distribuição normal das amostras, com média nula e desvio unitário. O código referente a padronização é mostrado na Figura 45 do Anexo B.

2.3.2 ANÁLISE DAS VARIÁVEIS

Após a filtragem e padronização os dados, foi analisada a distribuição das amostras para cada variável por meio de *Boxplots*, conforme mostra a Figura 10. Cada caixa representa a distribuição de uma variável descritas na Tabela 7, que são avaliadas por meio de cinco medidas de resumo: o mínimo, o primeiro quartil $Q1$, a mediana, o terceiro quartil $Q3$ e o máximo. A faixa de medição estendem-se até o menor e maior valor dentro de 1,5 vezes o intervalo interquartil (distância entre $Q1$ e $Q3$) a partir dos quartis. Pontos além da faixa de medição são considerados pontos fora da curva ou *outliers*, que são dados que diferem significativamente dos outros. Estes *outliers* em específico foram considerados na avaliação, pois eles passaram pelos filtros.

Figura 10 – *Bloxplots* das variáveis.



Fonte: Próprio autor.

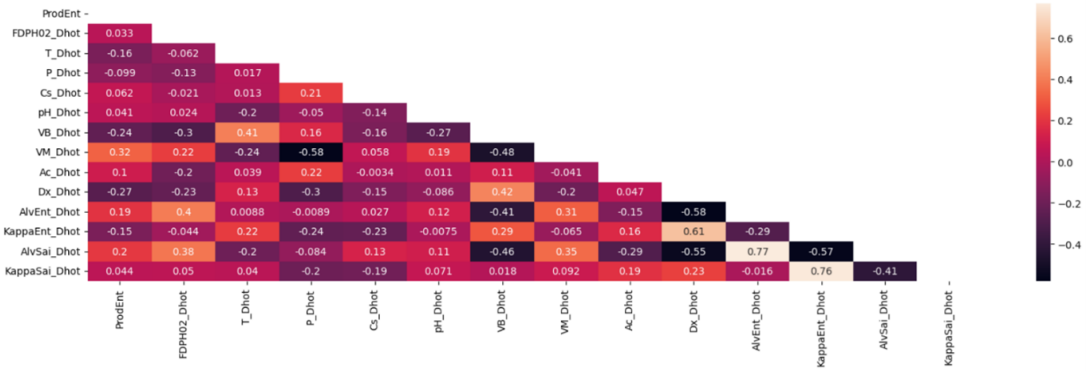
Tabela 7 – Relação das variáveis do boxplot.

Índice	**Variável -** D_{hot}
0	Produção Entrada
1	Fator de diluição da prensa Lavadora
2	Temperatura do estágio
3	Pressão do estágio
4	Consistência da polpa de celulose
5	pH da polpa de celulose
6	Vazão de vapor de baixa pressão
7	Vazão de vapor de média pressão
8	Vazão de ácido sulfúrico
9	Vazão de dióxido de cloro
10	Alvura da celulose na entrada do estágio
11	N° kappa da celulose na entrada do estágio

Fonte: Próprio autor.

A Figura 11 apresenta o *heatmap* das variáveis dependentes e independentes associadas ao problema. Ele permite visualizar a correlação entre as variáveis e identificar as que tem correlação mais significativa com as grandezas a serem preditas.

Figura 11 – *Heatmap* das variáveis dependentes e independentes.



Fonte: Próprio autor.

2.4 REDES NEURAIS ARTIFICIAIS

Uma RNA é estruturada como um modelo paramétrico, consistindo em unidades de processamento denominadas nós ou neurônios, que são organizados em camadas e conectados total ou parcialmente. Em particular, a rede neural do tipo MLP é composta por uma camada de entrada, que recebe os valores das variáveis, camadas ocultas ou intermediárias que processam os dados e uma camada de saída, que exibe o resultado. O número de neurônios, de camadas intermediárias e a função de ativação influenciam significativamente o desempenho dos modelos estabelecidos. O número de neurônios de saída, geralmente é igual ao número de variáveis dependentes em um problema de previsão baseado em uma relação de causa e efeito como é o caso desse trabalho. Abordagens heurísticas são geralmente usadas para determinar o número de neurônios ocultos (LIPPMANN, 1988), (BAILEY; THOMPSON, 1990), (MASTERS, 1993), (HAMZAÇEBI; AKAY; KUTAY, 2009).

A quantidade de nós nas camadas de entrada e saída é determinada pela natureza do problema em questão. O processo de definição do número de unidades ocultas envolve um procedimento experimental de busca. Após a definição da arquitetura da rede, incluindo o número de camadas e de unidades por camada, os pesos são ajustados para minimizar uma função de erro de predição, destacando a importância dos algoritmos de treinamento (HORNIK; STINCHCOMBE; WHITE, 1989).

As RNA exigem um volume significativo de dados para alcançar resultados confiáveis, o que demanda cautela no projeto e na validação das redes por meio de conjuntos de dados distintos. Em resumo, sua capacidade de mapear diretamente relações não lineares entre entrada e saída, junto à robustez e à flexibilidade para lidar com múltiplas entradas e saídas, consolidam as RNA como ferramentas valiosas para a modelagem de processos complexos.

Neste trabalho, duas redes do tipo MLP foram projetadas para cada estágio do branqueamento para realizarem a predição das variáveis *ALVURA* e NÚMERO *KAPPA*.

Os modelos MLP foram programados no *framework* do Keras e o ajuste dos hiperparâmetros foi realizado, utilizando o *Keras Tuner*. O código utilizado pode ser analisado a partir da Figura 47 do Anexo B.

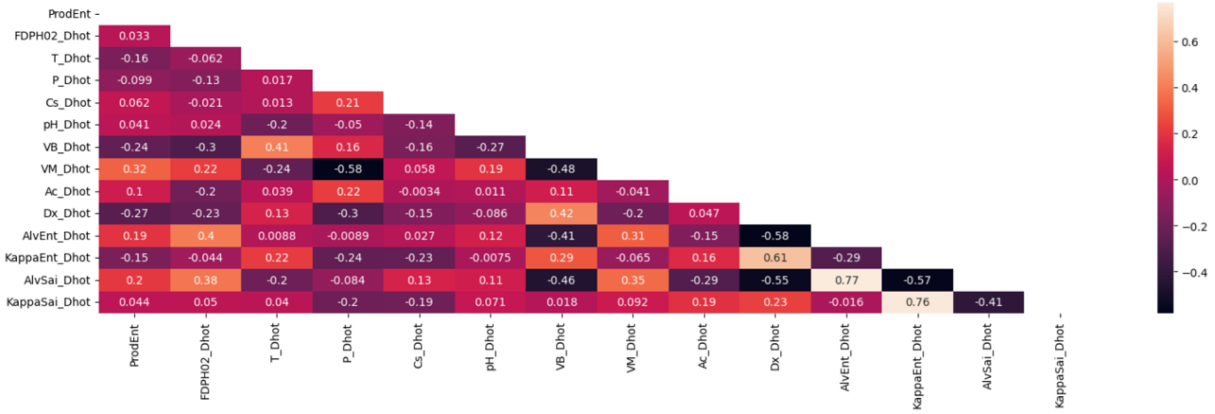
Este método permite a construção flexível de um modelo, no qual várias arquiteturas podem ser testadas automaticamente para determinar a configuração que melhor se adapta aos dados e ao problema em questão.

3 RESULTADOS

Neste estudo, focou-se na análise dos desempenhos alcançados pelas RNA na tarefa de predição das variáveis *ALVURA* e o NÚMERO *KAPPA*. Detalhou-se a metodologia empregada no projeto, treinamento e avaliação dessas redes, bem como foram exploradas as consequências práticas de seus desempenhos para a área de estudo em questão. Adicionalmente, realizou-se uma comparação crítica da eficácia destes modelos e delineou-se direções para investigações futuras no campo.

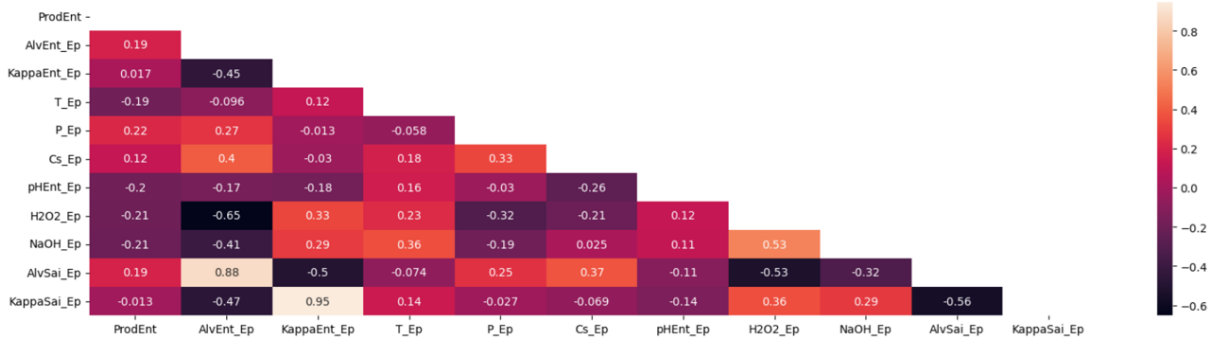
Na validação das variáveis de entrada (variáveis independentes), considerou-se todas as disponíveis na base de dados como sendo relevantes. Destaca-se, especialmente, as variáveis *'Temperatura', 'Pressão', 'pH' e 'Consistência'* da polpa no início de cada estágio do processo. A análise dessas variáveis através do *Heatmap* revelou uma correlação bastante reduzida com as variáveis de saída (variáveis dependentes). Isso se deve ao controle rigoroso desses parâmetros, que são ajustados automaticamente para operar dentro de uma faixa ideal pré-estabelecida, assegurando a eficiência das reações químicas em cada estágio. Consequentemente, suas variações são limitadas, resultando em pouca correlação com as variáveis de saída. Contudo, baseado em conhecimento técnico aprofundado, sabe-se que estas variáveis impactam significativamente a eficiência dos processos de branqueamento. Assim, decidiu-se por não excluí-las da análise. As Figuras 12, 13, 14 e 15 demonstram as correlações entre as variáveis de entrada e saída para cada estágio do branqueamento.

Figura 12 – Correlação das variáveis de entrada e saída - Estágio D_{hot}.



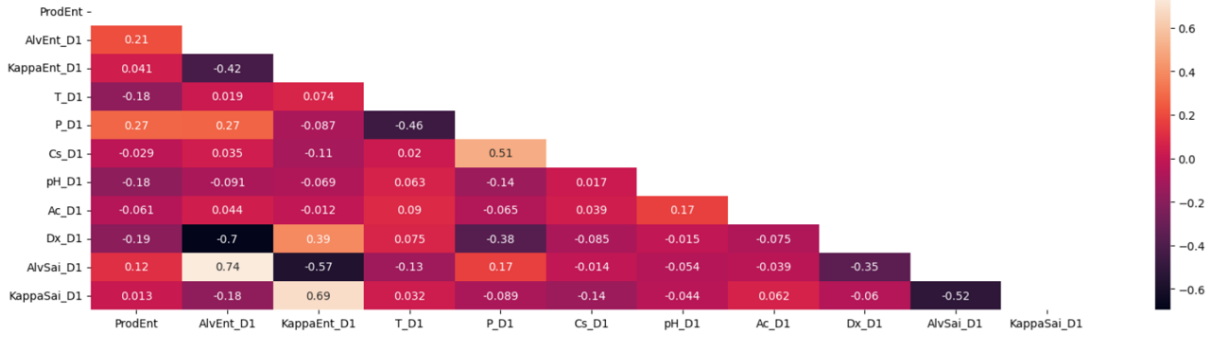
Fonte: Próprio autor.

Figura 13 – Correlação das variáveis de entrada e saída - Estágio E_p.



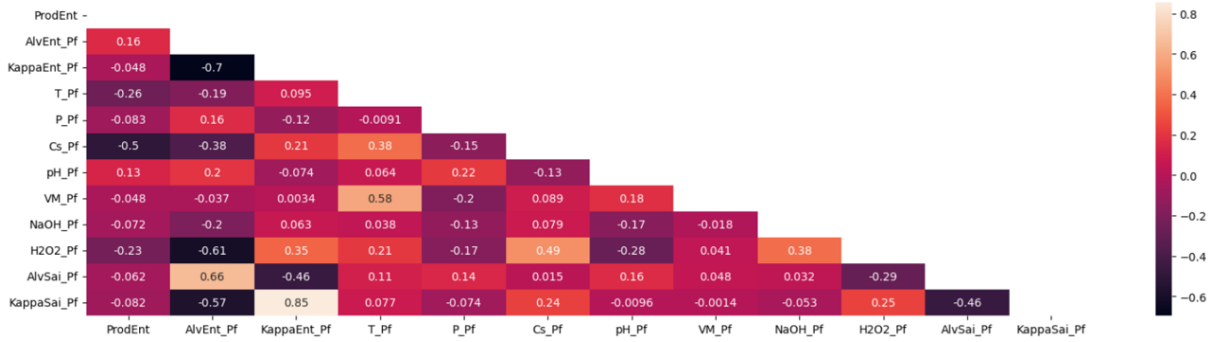
Fonte: Próprio autor.

Figura 14 – Correlação das variáveis de entrada e saída - Estágio D_1.



Fonte: Próprio autor.

Figura 15 – Correlação das variáveis de entrada e saída - Estágio P.



Fonte: Próprio autor.

A Tabela 8 apresenta o modelo da rede neural MLP implementada para a predição da *ALVURA* e NÚMERO *KAPPA* em cada estágio do branqueamento: D_{hot}, E_p, D_1 e P.

Na coluna *ALVURA*, as redes usam principalmente a função de ativação Tangente Hiperbólica (*tanh*) e nas camadas de entrada e ocultas e uma função de ativação *Linear* para a saída. As redes para a coluna NÚMERO *KAPPA* empregam a função de ativação *Rectified Linear Unit* (*ReLu*) em suas camadas de entrada e ocultas, também com uma saída *Linear*. O número de neurônios e a presença de camadas ocultas variam entre os estágios. Por exemplo, o estágio D_{hot} em *ALVURA* começa com 12 variáveis/neurônios na camada de entrada, tem duas camadas oculta com 4500 e 3920 neurônios e um neurônio na camada de saída. Na variável NÚMERO *KAPPA*, a MLP tem 12 variáveis/neurônios na camada de entrada, quatro camadas ocultas com 4500, 4890, 1210 e 1990 neurônios e 1 neurônio na camada de saída. As arquiteturas foram otimizadas para tarefas específicas ou conjuntos de dados associados a esses estágios, indicados pelo número variável de recursos de entrada e neurônios. A escolha de funções de ativação, como *tanh* e *ReLu*, pode afetar o aprendizado e o desempenho das redes de maneira diferente.

Tabela 8 – Estrutura das redes neurais MLP.

			ALVURA		**N° KAPPA**	
Estágio	**Rede neural**	**Nº variáveis de entrada**	**Nº neurônios**	**F. ativação**	**Nº neurônios**	**F. ativação**
D_{hot}	Camada Entrada	12	4500	tanh	4890	relu
	1ª Camada Oculta		3920	tanh	1210	relu
	2ª Camada Oculta		-	-	1990	relu
	Camada Saída		1	linear	1	linear
E_p	Camada Entrada	9	3270	tanh	1010	relu
	1ª Camada Oculta		2070	tanh	1010	relu
	2ª Camada Oculta		-	-	-	-
	Camada Saída		1	linear	1	linear
D_1	Camada Entrada	9	300	tanh	3930	relu
	1ª Camada Oculta		1320	tanh	3930	relu
	2ª Camada Oculta		4580	tanh	-	-
	Camada Saída		1	linear	1	linear
P	Camada Entrada	9	224	tanh	1500	relu
	1ª Camada Oculta		-	-	2590	relu
	2ª Camada Oculta		-	-	3830	relu
	3ª Camada Oculta		-	-	900	relu
	Camada Saída		1	linear	1	linear

Fonte: Próprio autor.

Os dados foram normalizados com a função *StandardScaler* e na sequência, a base de dados foi particionada na razão de 75% dos dados para o treinamento e 25% para a validação do modelo (Anexo B, Figuras 45 e 46).

Foi utilizada como ferramenta de avaliação do modelo, a média e o desvio padrão dos erros médios absolutos entre os valores preditos e os valores reais de *ALVURA* e NÚMERO *KAPPA* na saída de cada estágio de branqueamento, considerando 10 repetições para cada estágio. Para uma melhor visualização dos resultados também foi utilizado o coeficiente de determinação (R^2), com seus respectivos gráficos. Esses resultados estão detalhados nas Tabelas 9 a 12.

As redes neurais MLP foram capazes de fornecer uma predição acurada das variáveis apresentando um MAE médio variando entre 0,75 a 0,35 para a variável *ALVURA* e de 0,05 a 0,11 para a variável NÚMERO *KAPPA*. Quanto menor o valor do MAE, melhor o desempenho do modelo. Já a métrica R^2 indica o quanto da variabilidade dos dados é explicada pelo modelo, quanto mais próxima de 1 (ou 100%) melhor o ajuste do modelo de predição. Os valores médios de R^2 variaram entre 54% a 89% para *ALVURA* e entre 75% a 95% para NÚMERO *KAPPA*. Isto reforça a capacidade das redes neurais de fornecerem resultados confiáveis e com margem de erro reduzida.

Todos os códigos dos modelos da rede neural em cada estágio do branqueamento estão detalhadamente apresentados no Anexo B.

3.1 ESTÁGIO D_{hot}

Na Tabela 9 é mostrado o MAE do estágio D_{hot}. O MAE para a predição da *ALVURA* variou entre 0,723 e 0,807, enquanto para o NÚMERO *KAPPA* variou entre 0,098 e 0,136. Para a *ALVURA*, os valores de R^2 variaram entre 0,870 e 0,898, enquanto para o NÚMERO *KAPPA* variaram entre 0,808 e 0,868. Os resultados indicam que o desempenho do modelo é consistente. Para a previsão da *ALVURA*, o MAE médio foi de aproximadamente 0,754, o que indica que, em média, as previsões do modelo ficou a 0,754 unidades de distância dos valores reais. Além disso, o R^2 médio foi de aproximadamente 0,887, sugerindo que o modelo explicou cerca de 88,7% da variabilidade nos dados de *ALVURA*. Para o NÚMERO *KAPPA*, o MAE médio foi de aproximadamente 0,116 e o R^2 médio ficou em aproximadamente 0,839, o que indica um desempenho um pouco inferior em comparação com a previsão da *ALVURA*, porém ainda satisfatório considerando dados industriais. Os valores do desvio padrão do MAE e R^2 sugerem que os resultados foram estáveis entre as repetições do experimento.

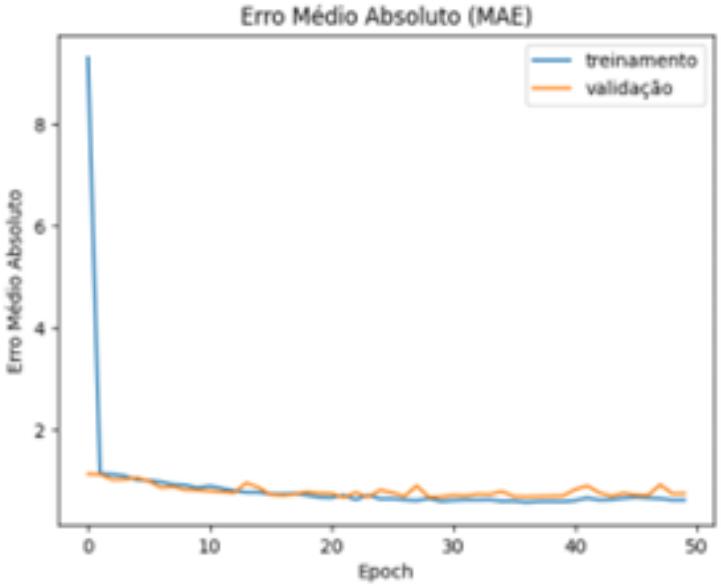
Os valores de MAE e R^2 indicam que o modelo foi capaz de capturar uma quantidade significativa da variabilidade nos dados, o que é encorajador. Como exemplo, os gráficos mostrados nas Figuras 16 a 21 foram resultados obtidos pelos respectivos modelos gerados na décima repetição exemplificado na Tabela 9.

Tabela 9 – MAE e R^2 obtido em cada iteração para as variáveis *ALVURA* e NÚMERO *KAPPA* - Estágio D_{hot}.

Repetições	ALVURA		Nº KAPPA	
	MAE	R^2	MAE	R^2
1	0,772	0,887	0,106	0,850
2	0,741	0,882	0,118	0,837
3	0,723	0,896	0,135	0,822
4	0,780	0,872	0,107	0,853
5	0,757	0,892	0,108	0,842
6	0,807	0,870	0,098	0,868
7	0,756	0,892	0,136	0,821
8	0,747	0,890	0,116	0,844
9	0,731	0,898	0,107	0,848
10	0,731	0,892	0,133	0,808
Média	0,754	0,887	0,116	0,839
Desvpad	0,026	0,010	0,014	0,018

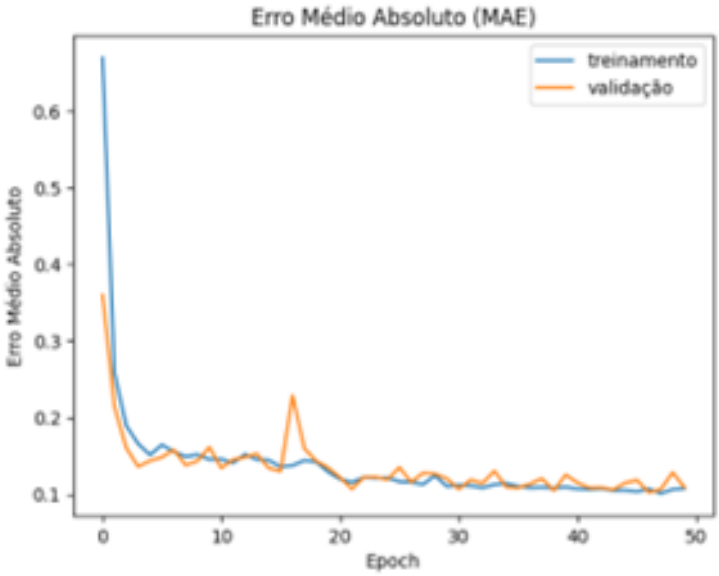
Fonte: Cenibra.

Figura 16 – Valor do MAE para a *ALVURA* durante as épocas de iteração com os dados de treinamento e teste - Estágio D_{hot}.



Fonte: Próprio autor.

Figura 17 – Plotagem dos valores de MAE para o NÚMERO *KAPPA* para os dados de teste e treinamento - Estágio D_{hot}.



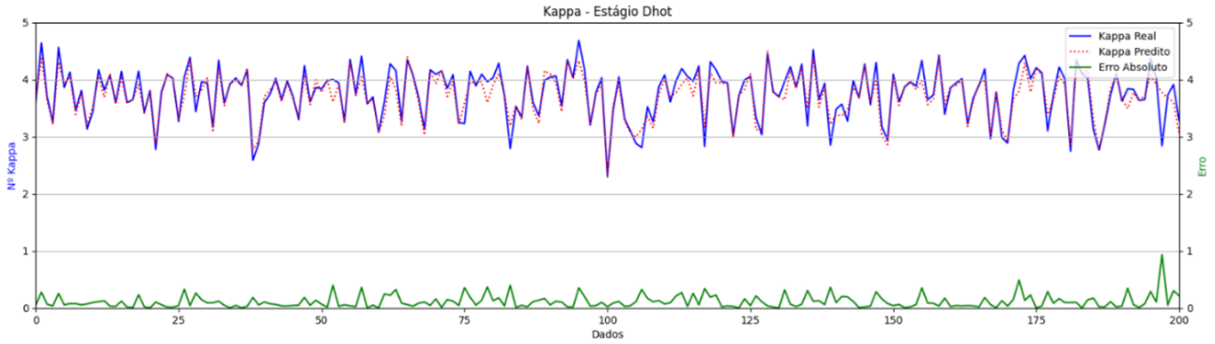
Fonte: Próprio autor.

As Figuras 18 e 19 mostram o acompanhamento dos dados reais em comparação aos preditos e o erro associado para os 200 primeiros pontos, considerando as variáveis *ALVURA* e o NÚMERO *KAPPA*, repectivamente. Elas possibilitam a visualização da proximidade entre as curvas para os dados reais e preditos e indicando a eficácia do modelo em realizar previsões precisas.

Figura 18 – Valores da *ALVURA* predita vs. real e o erro com dados de teste - Estágio D_{hot}.

Fonte: Próprio autor.

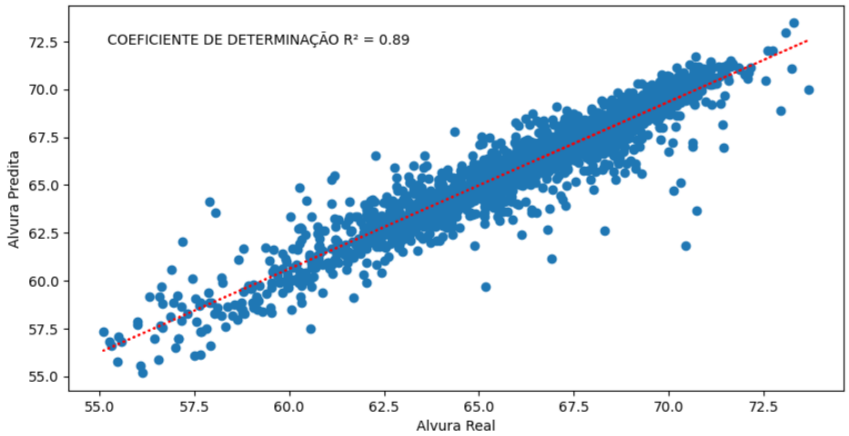
Figura 19 – Valores de NÚMERO *KAPPA* predito vs. real e o erro com dados de teste - Estágio D_{hot}.



Fonte: Próprio autor.

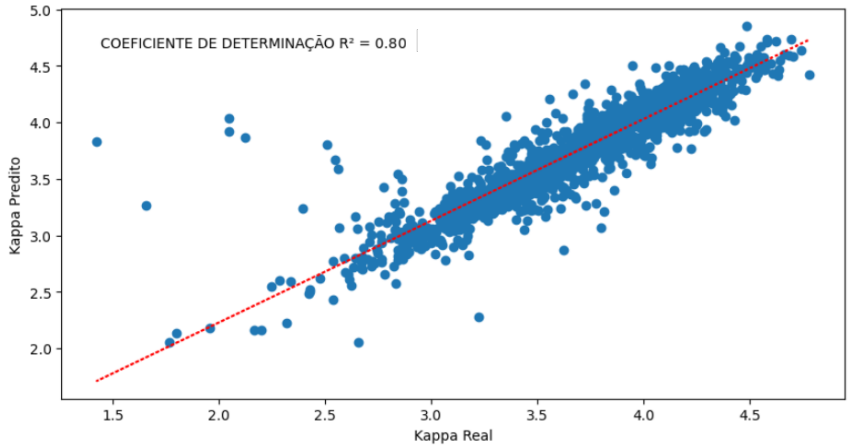
Uma alternativa para visualizar os resultados da predição é através do gráfico de dispersão, que incorpora a linha de regressão linear e apresenta o Coeficiente de Determinação calculado. As Figuras 20 e 21 exibem estas informações relevantes de maneira clara e integrada e o valor de R^2 entre a predição e os dados reais para as variáveis *ALVURA* e NÚMERO *KAPPA*, respectivamente.

Figura 20 – Curva de dispersão da variável número *ALVURA* considerando os valores predito vs. real da base de dados de teste - Estágio D_{hot}.



Fonte: Próprio autor.

Figura 21 – Curva de dispersão da variável NÚMERO *KAPPA* considerando os valores predito vs. real da base de dados de teste - Estágio D_{hot}.



Fonte: Próprio autor.

3.2 ESTÁGIO E_p

A Tabela 10 fornece um relatório de métricas de desempenho para o modelo de predição associado a duas características principais: *ALVURA* e o NÚMERO *KAPPA*. Os valores de MAE variaram entre aproximadamente 0,643 e 0,775 para a *ALVURA* e para o NÚMERO *KAPPA*, variaram entre 0,039 e 0,063.

Avaliando a variabilidade dos dados para a *ALVURA*, os valores de R^2 variaram em torno de 0,802 e 0,868 e para o NÚMERO *KAPPA*, variaram entre 0,934 e 0,960. O MAE médio das dez medições para a *ALVURA* foi por volta de 0,692, indicando que as previsões do modelo estavam a 0,692 unidades de distância, em média, dos valores reais de *ALVURA*. Além disso, o R^2 médio das medições foi de aproximadamente 0,839, sugerindo que o modelo explicou cerca de 83,9% da variabilidade nos dados de *ALVURA*.

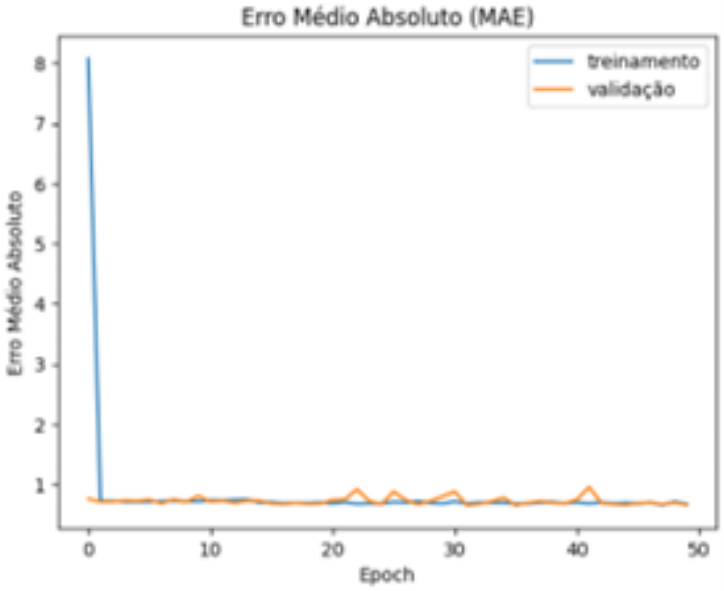
Para o NÚMERO *KAPPA*, o MAE médio foi de aproximadamente 0,046 e o R^2 médio foi em torno de 0,950, indicando um desempenho geralmente bom na previsão do NÚMERO *KAPPA*. A consistência dos resultados da Tabela 10, o desvio padrão dos valores de MAE e R^2 também foi relativamente baixo, sugerindo uma consistência nos resultados entre as épocas de treinamento. Esses resultados sugerem que o modelo teve um desempenho razoavelmente bom na previsão tanto da *ALVURA* quanto do NÚMERO *KAPPA*. No entanto, é importante continuar avaliando e ajustando o modelo conforme necessário para melhorar ainda mais sua precisão e generalização. Os gráficos mostrados nas Figuras 22 a 27 foram resultados obtidos pelos respectivos modelos gerados na décima repetição exemplificado na Tabela 10.

Tabela 10 – MAE e R^2 obtido em cada iteração para as variáveis *ALVURA* e NÚMERO *KAPPA* - Estágio E_p.

Repetições	ALVURA		Nº KAPPA	
	MAE	R^2	MAE	R^2
1	0,643	0,859	0,048	0,950
2	0,658	0,847	0,042	0,956
3	0,678	0,817	0,063	0,934
4	0,671	0,861	0,049	0,947
5	0,775	0,802	0,040	0,960
6	0,648	0,840	0,049	0,946
7	0,723	0,812	0,039	0,957
8	0,663	0,851	0,046	0,951
9	0,702	0,868	0,043	0,951
10	0,756	0,834	0,045	0,950
Média	0,692	0,839	0,046	0,950
Desvpad	0,046	0,022	0,007	0,007

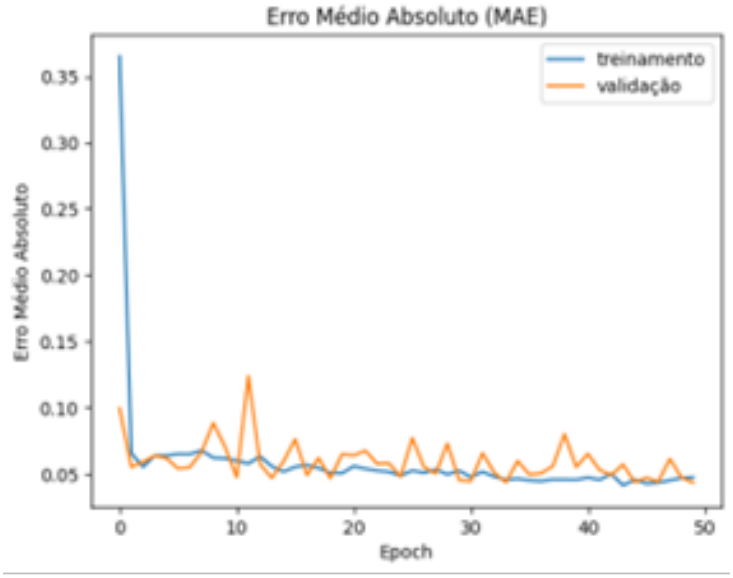
Fonte: Cenibra.

Figura 22 – Plotagem dos valores de MAE para a variável *ALVURA* considerando os dados de teste e treinamento - Estágio E_p.



Fonte: Próprio autor.

Figura 23 – Plotagem dos valores de MAE para a variável NÚMERO *KAPPA* considerando os dados de treinamento e teste - Estágio E_p.

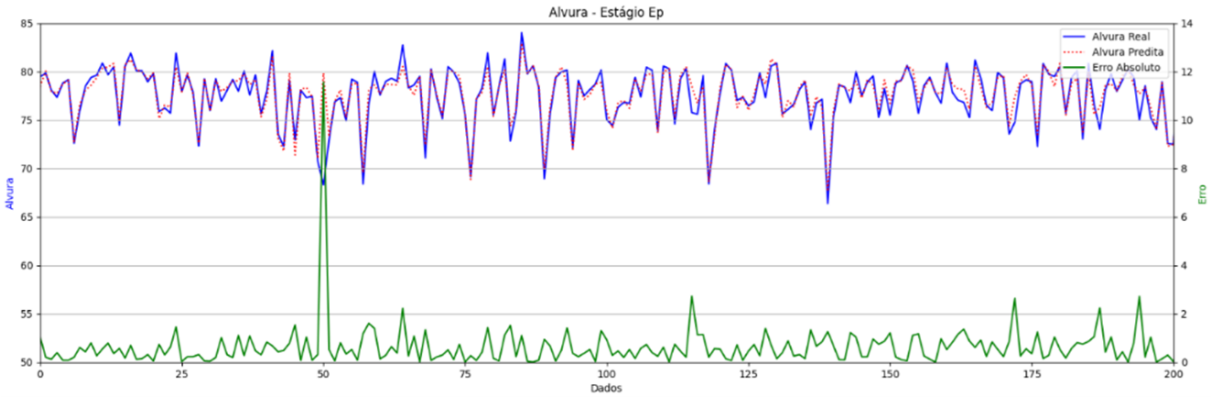


Fonte: Próprio autor.

As Figuras 24 e 25 mostram o acompanhamento dos dados reais em comparação aos preditos e o erro associado para os 200 primeiros pontos, considerando as variáveis *ALVURA* e NÚMERO *KAPPA*, respectivamente. Elas possibilitam a visualização da proximidade entre as curvas para os dados reais e preditos e indicando a eficácia do modelo em realizar previsões precisas.

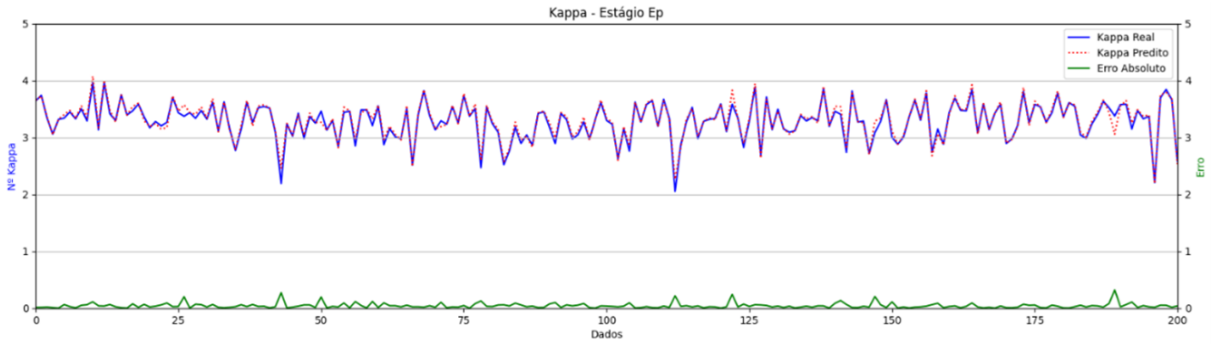
O pico em torno do ponto 50 na Figura 24 é particularmente alto, indicando uma discrepância significativa entre a *ALVURA* real e a predita. Pode ser resultado de uma anomalia nos dados, uma falha temporária no processo de medição ou um evento atípico que não foi bem capturado pelo modelo preditivo. Também se observam outros picos menores em outras amostras, que podem ser devidos a variabilidades normais no processo ou a limitações no modelo preditivo. Identificar e analisar as causas desses picos pode ajudar a melhorar o modelo preditivo e o processo de medição de alvura. Pode ser útil investigar os dados e o processo em torno dos pontos de maior erro para entender se há fatores específicos que estão contribuindo para essas discrepâncias.

Figura 24 – Valores da *ALVURA* predita vs. real e MAE considerando os dados de teste - Estágio E_p.



Fonte: Próprio autor.

Figura 25 – Valores de NÚMERO *KAPPA* predito vs. real e MAE considerando os dados de teste - Estágio E_p.

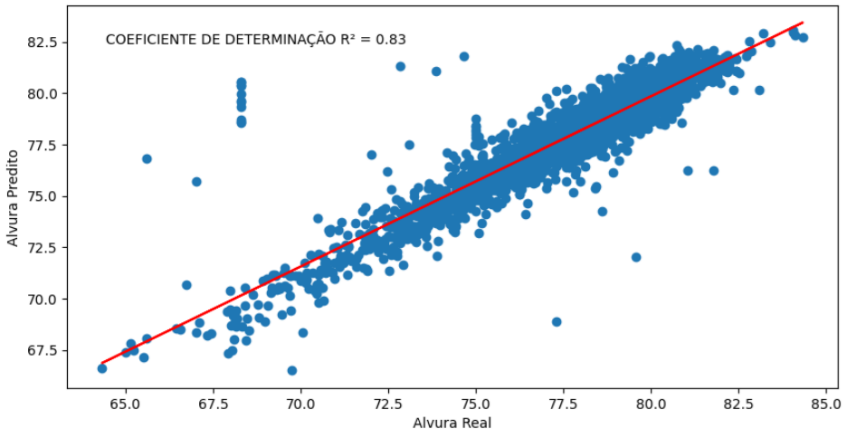


Fonte: Próprio autor.

As Figuras 26 27 mostram os gráficos de dispersão complementado pela linha de regressão linear e o valor de R^2 calculado, para as variáveis *ALVURA* e NÚMERO *KAPPA*, respectivamente. Estes gráficos destacam de maneira clara e concisa todas as informações cruciais do estudo.

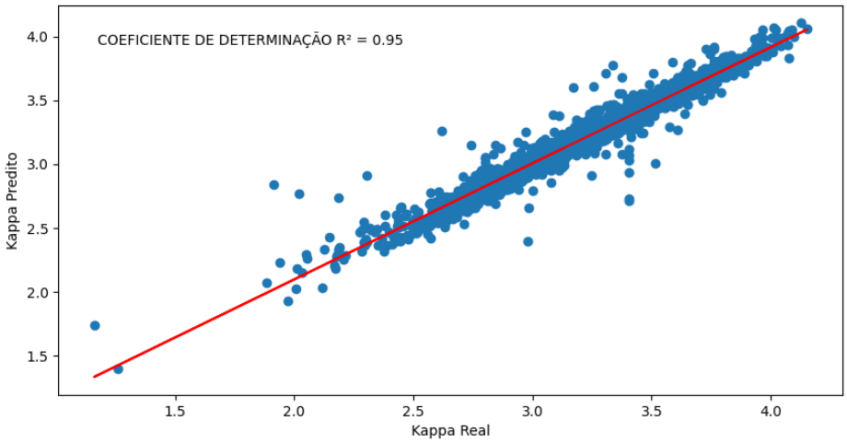
Vale ressaltar o alto R^2 de 0.95 obtido para a variável NÚMERO *KAPPA*, mostrando que o ajuste do modelo entre os valores preditos e reais foi muito eficiente.

Figura 26 – Curva de dispersão da variável número *ALVURA* considerando os valores predito vs. real da base de dados de teste - Estágio E_p.



Fonte: Próprio autor.

Figura 27 – Curva de dispersão da variável NÚMERO *KAPPA* considerando os valores predito vs. real da base de dados de teste - Estágio E_p.



Fonte: Próprio autor.

Esse procedimento é um padrão na avaliação de modelos de aprendizado de máquina. Monitorar o MAE durante o treinamento e a validação fornece *insights* sobre a qualidade das previsões do modelo. Idealmente, deseja-se um valor de MAE baixo e similar nos conjuntos de treinamento e de validação. Divergências grandes entre as métricas de treinamento e validação podem indicar *overfitting* o que não foi o caso. Visualizar essas métricas ao longo do tempo ajuda a ajustar os *hiperparâmetros*, como o número de épocas, para melhorar o desempenho do modelo antes de usá-lo para fazer previsões reais.

3.3 ESTÁGIO D_1

A Tabela 11 mostrada a seguir fornece as duas métricas estatísticas, o MAE e o R^2 para duas variáveis *ALVURA* e NÚMERO *KAPPA*. Os resultados para *ALVURA* mostram que os valores de MAE variaram entre 0,528 e 0,679 durante as iterações, apresentando uma média de 0,573 e um desvio padrão de 0,048.

Os valores de R^2 para *ALVURA* variaram em uma faixa mais estreita (de 0,648 a 0,785), com uma média em torno de 0,738 e um desvio padrão de 0,041.

Para o NÚMERO *KAPPA*, os valores de MAE ficaram na faixa de 0,084 a 0,093, com uma média de aproximadamente 0,089 e um desvio padrão muito baixo de 0,003, o que indicou uma precisão muito consistente nas previsões. O R^2 variou entre 0,739 e 0,765, com uma média próxima a 0,754 e um desvio padrão de 0,010. Estes resultados sugeriram que o modelo da variável NÚMERO *KAPPA* tem uma boa capacidade para explicar a variação da variável dependente.

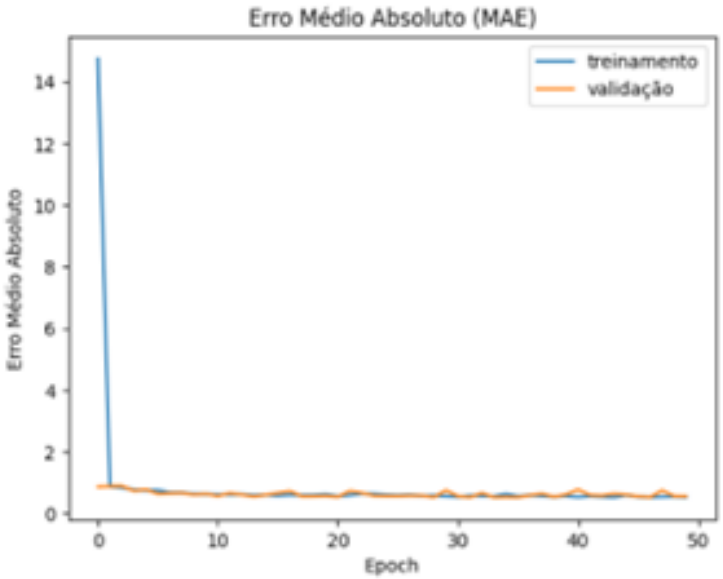
As Figuras 28 a 33 apresentam os resultados pelos respectivos modelos gerados na décima repetição exemplificada na Tabela 11.

Tabela 11 – MAE e R^2 obtido em cada iteração para as variáveis *ALVURA* e NÚMERO *KAPPA* - Estágio D_1.

Repetições	ALVURA		Nº KAPPA	
	MAE	R^2	MAE	R^2
1	0,535	0,737	0,089	0,761
2	0,528	0,773	0,091	0,739
3	0,541	0,778	0,087	0,762
4	0,575	0,728	0,086	0,764
5	0,679	0,648	0,091	0,749
6	0,611	0,701	0,093	0,740
7	0,580	0,748	0,089	0,750
8	0,535	0,785	0,089	0,744
9	0,538	0,757	0,085	0,761
10	0,607	0,727	0,084	0,765
Média	0,573	0,738	0,089	0,754
Desvpad	0,048	0,041	0,003	0,010

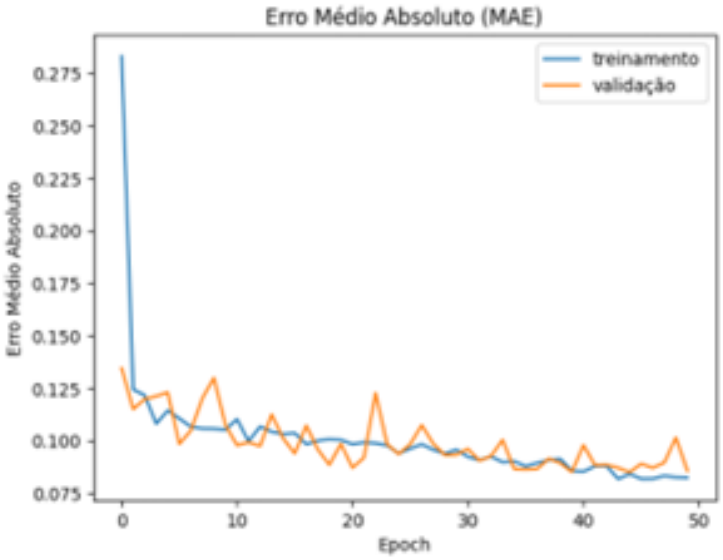
Fonte: Cenibra.

Figura 28 – Valores de MAE da variável *ALVURA* considerando os dados de treinamento e teste - Estágio D_1.



Fonte: Próprio autor.

Figura 29 – Valores de MAE da variável NÚMERO *KAPPA* considerando os dados de treinamento e teste - Estágio D_1.



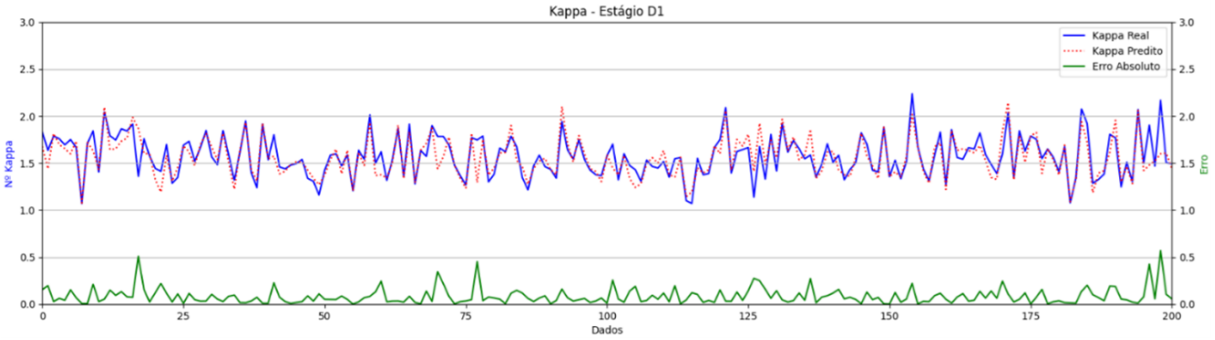
Fonte: Próprio autor.

As Figuras 30 e 31 ilustram o acompanhamento dos dados reais em comparação aos preditos e o erro associado para os 200 primeiros pontos considerando as variáveis *ALVURA* e NÚMERO *KAPPA*, respectivamente. Elas possibilitam a visualização da proximidade entre as curvas para os dados reais e preditos e indicando a eficácia do modelo em realizar previsões precisas.

Figura 30 – Valores da *ALVURA* predita vs. real e o MAE considerando os dados de teste - Estágio D_1.

Fonte: Próprio autor.

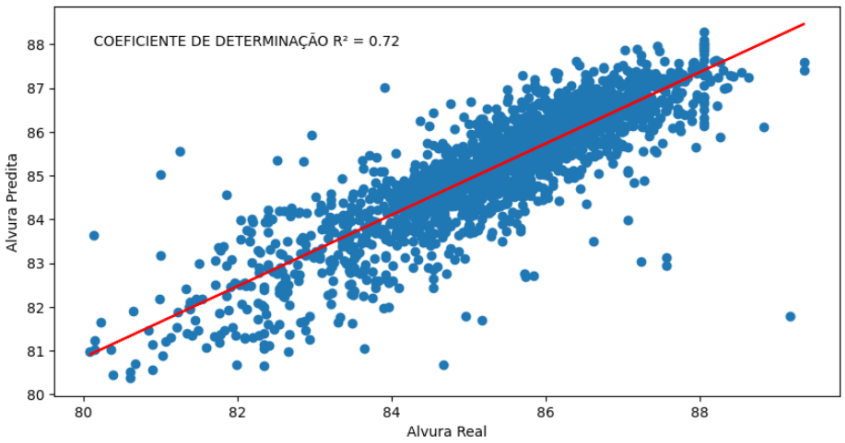
Figura 31 – Valores de NÚMERO *KAPPA* predito vs. real e MAE considerando os dados de teste - Estágio D_1.



Fonte: Próprio autor.

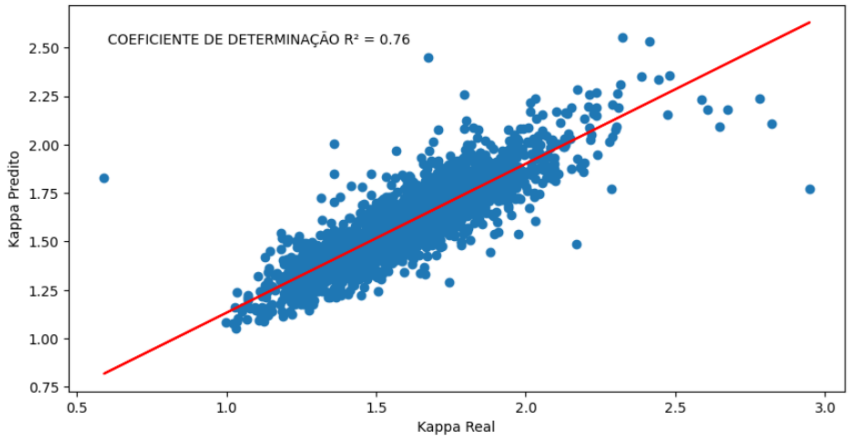
O gráfico de dispersão dos dados com a linha de tendência e o valor de R^2 é apresentado nas Figuras 32 e 33 para as variáveis *ALVURA* e NÚMERO *KAPPA*, respectivamente.

Figura 32 – Curva de dispersão da variável número *ALVURA* considerando os valores predito vs. real da base de dados de teste - Estágio D_1.



Fonte: Próprio autor.

Figura 33 – Curva de dispersão da variável NÚMERO *KAPPA* considerando os valores predito vs. real da base de dados de teste - Estágio D_1.



Fonte: Próprio autor.

3.4 ESTÁGIO *P*

A Tabela 12 mostra os resultados para 10 repetições, o que pode indicar a performance do modelo ou a consistência da medição ao longo do tempo ou em diferentes amostras. Para o MAE de aproximadamente 0,331 a 0,371, com uma média de cerca de 0,348 e um desvio padrão de 0,014 sugerem que as previsões para *ALVURA* são tanto precisas quanto consistentes. Por outro lado, os valores do coeficiente de determinação (R^2) mostram uma variação maior, oscilando entre 0,495 e 0,598, com uma média aproximada de 0,545 e um desvio padrão de 0,036.

Isso indica que a capacidade de *ALVURA* para explicar a variação da variável dependente apresenta uma variabilidade relativamente maior em comparação com a precisão das previsões.

No caso do NÚMERO *KAPPA*, os valores de MAE são igualmente baixos, variando de aproximadamente 0,072 a 0,096, com uma média de cerca de 0,078 e um desvio padrão de 0,008. Essa faixa é um pouco mais ampla em comparação com a variável *ALVURA*. Em contraste, o coeficiente de determinação (R^2) para o NÚMERO *KAPPA* demonstra ser alto e mais consistente, oscilando entre cerca de 0,73 a 0,83, com uma média próxima de 0,80 e um desvio padrão de 0,033. Estes resultados indicam que o NÚMERO *KAPPA* é uma variável ou método com uma capacidade robusta e consistente de explicar a variação da variável dependente.

Em comparação com dados anteriores, observa-se que o NÚMERO *KAPPA* frequentemente apresenta valores mais elevados do coeficiente de determinação (R^2), sugerindo que ele pode ser um indicador mais eficaz para previsões ou para explicar a variabilidade da variável dependente do que *ALVURA*. No entanto, é importante salientar que o valor de R^2 não deve ser o único critério para avaliar a adequação de um modelo.

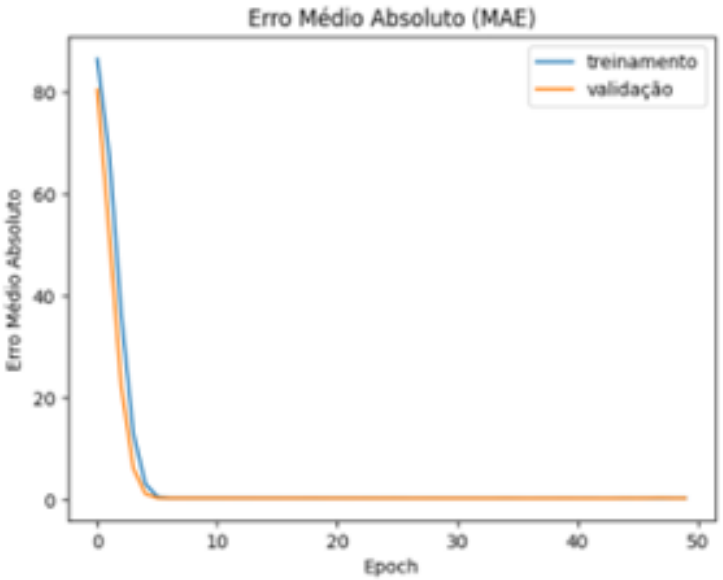
A precisão das previsões, medida pelo MAE, e a relevância do modelo em relação ao fenômeno ou questão estudada também são aspectos cruciais. Os gráficos mostrados nas Figuras 38 a 39 foram oriundos da décima repetição do modelo exemplificado na Tabela 12.

Tabela 12 – MAE e R^2 obtido em cada iteração para as variáveis *ALVURA* e NÚMERO *KAPPA* - Estágio *P*.

Repetições	ALVURA		Nº KAPPA	
	MAE	R^2	MAE	R^2
1	0,371	0,498	0,075	0,810
2	0,358	0,558	0,075	0,818
3	0,356	0,537	0,076	0,806
4	0,328	0,583	0,096	0,725
5	0,346	0,510	0,072	0,833
6	0,334	0,579	0,090	0,756
7	0,362	0,542	0,074	0,812
8	0,352	0,495	0,075	0,817
9	0,344	0,537	0,076	0,812
10	0,331	0,598	0,076	0,806
Média	0,348	0,544	0,078	0,799
Desvpad	0,014	0,036	0,008	0,033

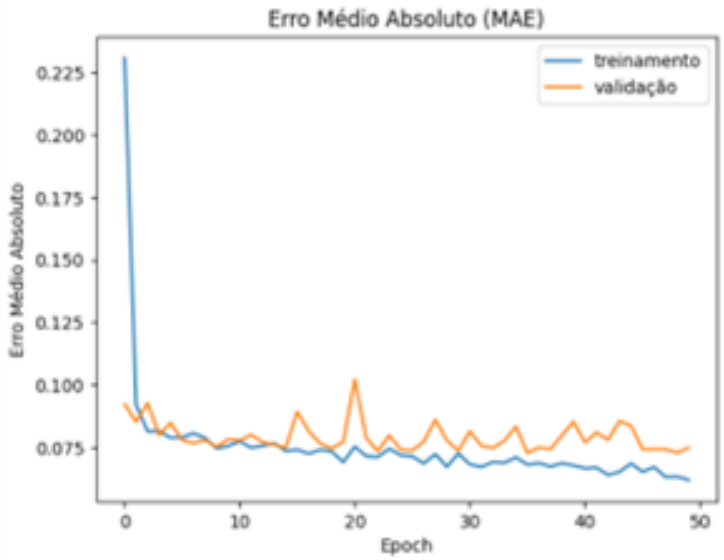
Fonte: Cenibra.

Figura 34 – Valores do MAE para a variável *ALVURA* considerando os dados de treinamento e teste - Estágio *P*.



Fonte: Próprio autor.

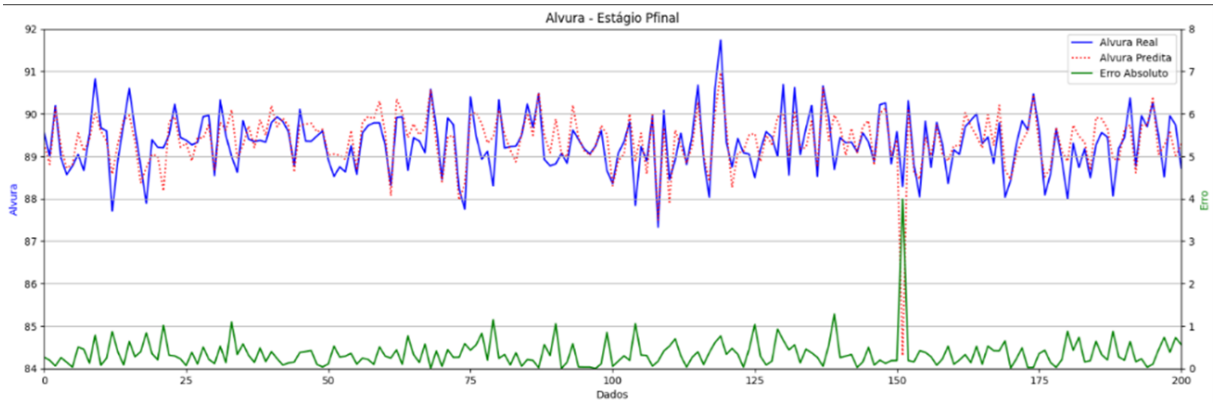
Figura 35 – Valores do MAE para a variável NÚMERO *KAPPA* considerando os dados de treinamento e teste - Estágio *P*.



Fonte: Próprio autor.

As Figuras 36 e 37 ilustram o acompanhamento dos dados reais em comparação aos preditos e o erro associado para os 200 primeiros pontos das variáveis *ALVURA* e NÚMERO *KAPPA*, respectivamente. A proximidade entre as curvas dos valores reais e preditos é notavelmente evidente, destacando a alta eficácia do modelo na tarefa de predição. Este alinhamento sublinha a precisão do modelo em replicar as tendências e variações dos dados reais e mostra a precisão do modelo.

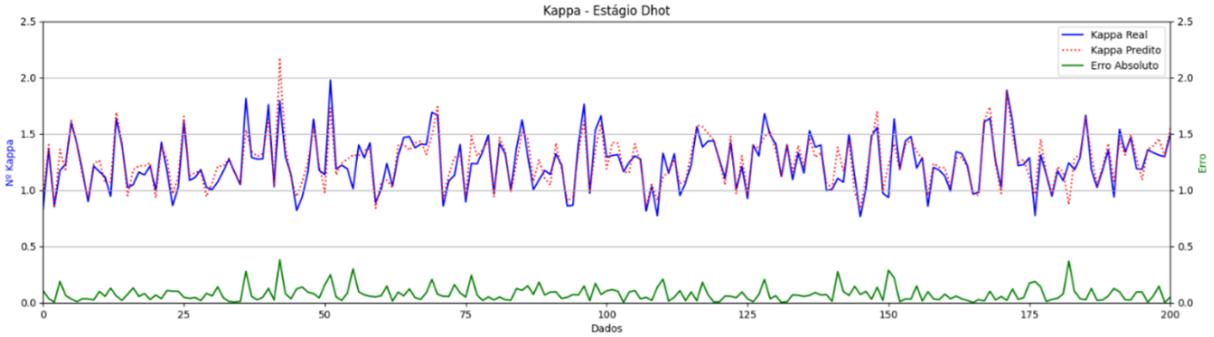
Figura 36 – Valores da *ALVURA* predita vs. real e MAE considerando os dados de teste - Estágio *P*.



Fonte: Próprio autor.

Em torno do da amostra 150 da Figura 36 se observa um MAE maior para a *ALVURA*. Este pico pode ser resultado de um evento anômalo, como uma alteração inesperada no processo de produção ou uma falha temporária no sistema de medição ou predição. Há ocorrência de outros picos menos pronunciados ao longo deste gráfico. Esses picos menores podem ser devidos a flutuações normais no processo de produção que o modelo preditivo não conseguiu capturar perfeitamente.

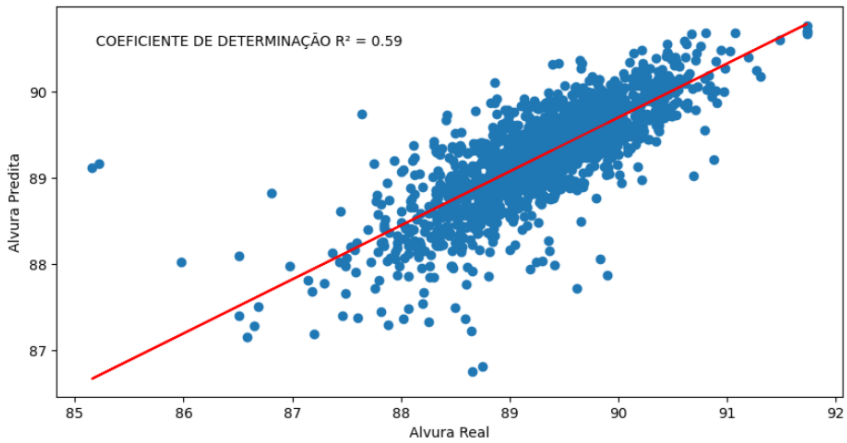
Figura 37 – Valores de NÚMERO *KAPPA* predito vs. real e MAE considerando os dados de teste - Estágio *P*.



Fonte: Próprio autor.

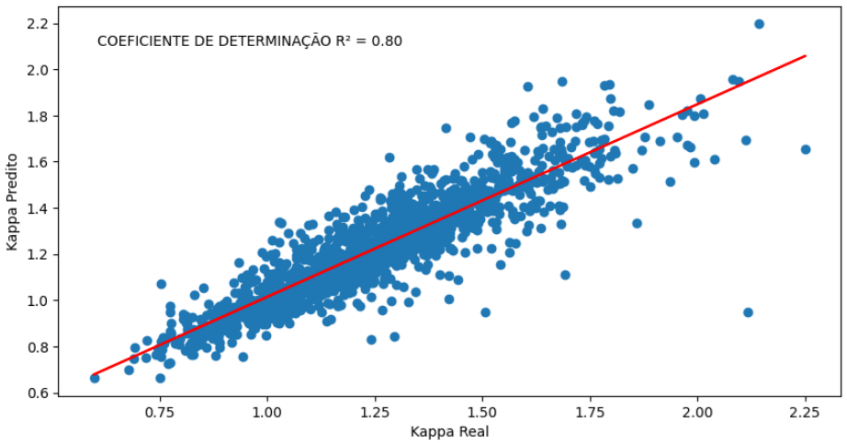
Os gráficos de dispersão apresentados nas Figuras 38 e 39 mostram o R^2 entre os dados reais e preditos para as variáveis *ALVURA* e NÚMERO *KAPPA*, respectivamente. Nestas figuras também são apresentados o valor de R^2 e a linha de regressão linear. Eles proporcionam uma visualização integrada das informações úteis e permitem verificar a precisão dos modelos preditivos.

Figura 38 – Curva de dispersão da variável *ALVURA* considerando os valores predito vs. real da base de dados de teste - Estágio *P*.



Fonte: Próprio autor.

Figura 39 – Curva de dispersão da variável NÚMERO *KAPPA* considerando os valores predito vs. real da base de dados de teste - Estágio *P*.



Fonte: Próprio autor.

4 CONCLUSÃO

Ao longo deste trabalho foram implementados, detalhados e avaliados um total de oito modelos preditivos usando as redes neurais do tipo MLP. Cada modelo foi ajustado de modo a realizar a predição das variáveis *ALVURA* e número *KAPPA* em cada um dos quatro estágios do branqueamento: D_{hot}, E_p, D_1 e P. As métricas de avaliação adotadas para todos os modelos foram o MAE e o R^2.

No âmbito do modelo *ALVURA*, observou-se uma notável consistência nos valores de MAE, os quais variaram em um espectro limitado, com uma média que sugeriu um equilíbrio entre a acurácia e a ocorrência de erros. O R^2 manteve-se em um intervalo superior, denotando uma robustez preditiva que, apesar de não perfeita, foi certamente competente. Os desvios padrão, tanto para o MAE quanto para o R^2, foram contidos, o que reforça a confiabilidade das previsões do modelo *ALVURA* ao longo das iterações.

Por outro lado, o modelo NÚMERO *KAPPA* apresentou uma gama mais ampla de resultados para o MAE, com uma média que, embora ligeiramente superior ao da *ALVURA*, ainda indicou um desempenho satisfatório. O R^2 para o modelo NÚMERO *KAPPA* foi notavelmente robusto, refletindo uma capacidade preditiva de alto calibre, com médias e desvios padrão que atestaram a sua consistência e confiabilidade em diferentes contextos.

As médias e os desvios padrão analisados revelaram que, embora o modelo *ALVURA* seja confiável e consistente, foi o modelo NÚMERO *KAPPA* que se sobressaiu em precisão preditiva e estabilidade de resultados. Isto foi particularmente evidente ao considerarmos o R^2 mais próximo de 1 e os menores desvios padrão associados ao modelo NÚMERO *KAPPA*. Diante dos dados comparativos, a escolha entre os modelos depende do peso que cada métrica tem para a aplicação na planta de branqueamento da celulose. No entanto, de maneira geral, o modelo NÚMERO *KAPPA* mostrou-se como a opção mais viável, na prática industrial, dada a sua previsão acurada e consistente.

Referências

AGUIAR, H. C.; FILHO, R. M. Neural network and hybrid model: a discussion about different modeling techniques to predict pulping degree with industrial data. **Chemical engineering science**, Elsevier, v. 56, n. 2, p. 565–570, 2001.

AL-AWAMI, L.; SIDRAK, Y. Understanding the dynamic behavior of kamyr digesters. **ISA transactions**, Elsevier, v. 37, n. 1, p. 53–64, 1998.

BAILEY, D.; THOMPSON, D. How to develop neural-network applications. **AI expert**, Miller Freeman, Inc. Lawrence, KS, USA, v. 5, n. 6, p. 38–47, 1990.

BAJPAI, P. **Green chemistry and sustainability in pulp and paper industry**. [S.l.]: Springer, 2015.

BRUMANO, P. **Produção de Celulose e Sistema Kraft - TermoBlog — termoblog.com.br**. 2023. <https://termoblog.com.br/producao-de-celulose-e-sistema-kraft/>. [Accessed 25-04-2024].

CANAKCI, A.; OZSAHIN, S.; VAROL, T. Modeling the influence of a process control agent on the properties of metal matrix composite powders using artificial neural networks. **Powder technology**, Elsevier, v. 228, p. 26–35, 2012.

CARDOSO, G. V.; FRIZZO, S. M. B.; ROSA, C.; FOELKEL, C. E. B.; ASSIS, T.; OLIVEIRA, P. Otimização das condições do cozimento kraft de eucalyptus globulus em função do teor de lignina da madeira. In: **35º Congresso e Exposição Anual de Celulose e Papel, Oct**. [S.l.: s.n.], 2002. p. 14–17.

CARVALHO, F. B. d.; TORRES, B. S.; FONSECA, M. d. O.; FILHO, C. S. Sistemas pims-conceituação, usos e benefícios. **Tecnologia em Metalurgia, Materiais e Mineração**, ABM-Associação Brasileira de Metalurgia, Materiais e Mineração, v. 1, n. 4, p. 1–5, 2013.

CEYLAN, I. Determination of drying characteristics of timber by using artificial neural networks and mathematical models. **Drying Technology**, Taylor & Francis, v. 26, n. 12, p. 1469–1476, 2008.

COMELATO, J. S. Efeito de reagentes de branqueamento nas propriedaes físicas e mecânicas da polpa de celulose kraft de eucalipto. Universidade Estadual Paulista (Unesp), 2011.

DUFOUR, P.; BHARTIYA, S.; DHURJATI, P. S.; III, F. J. D. Neural network-based software sensor: training set design and application to a continuous pulp digester. **Control Engineering Practice**, Elsevier, v. 13, n. 2, p. 135–143, 2005.

ESTEBAN, L. G.; FERNÁNDEZ, F. G.; PALACIOS, P. de. Prediction of plywood bonding quality using an artificial neural network. Walter de Gruyter, 2011.

FERNANDES, N. C.; CASTRO, J. A. Steady-state simulation of a continuous moving bed reactor in the pulp and paper industry. **Chemical engineering science**, Elsevier, v. 55, n. 18, p. 3729–3738, 2000.

FERREIRA, C. R. d. S. Otimização do perfil de temperatura na polpação rdh de eucalyptus sp. Universidade Federal de Viçosa, 2000.

GARDNER, M. W.; DORLING, S. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. **Atmospheric environment**, Elsevier, v. 32, n. 14-15, p. 2627–2636, 1998.

GELLERSTEDT, G.; PRANDA, J.; LINDFORS, E.-L. Structural and molecular properties of residual birch kraft lignins. **Journal of Wood Chemistry and Technology**, Taylor & Francis, v. 14, n. 4, p. 467–482, 1994.

HAMZAÇEBI, C.; AKAY, D.; KUTAY, F. Comparison of direct and iterative artificial neural network forecast approaches in multi-periodic time series forecasting. **Expert systems with applications**, Elsevier, v. 36, n. 2, p. 3839–3844, 2009.

HARKONEN, E. J. A mathematical model for two-phase flow in a continuous digester. **Tappi journal**, v. 70, n. 12, p. 122–126, 1987.

HIMMELBLAU, D. M. Applications of artificial neural networks in chemical engineering. **Korean journal of chemical engineering**, Springer, v. 17, p. 373–392, 2000.

HORNIK, K.; STINCHCOMBE, M.; WHITE, H. Multilayer feedforward networks are universal approximators. **Neural networks**, Elsevier, v. 2, n. 5, p. 359–366, 1989.

III, F. J. D.; KAYIHAN, F. Reaction profile control of the continuous pulp digester. **Chemical engineering science**, Elsevier, v. 54, n. 13-14, p. 2679–2688, 1999.

KAYIHAN, F. **A stochastic continuous digester model to capture transition, compaction and chip size distribution effects**. 2002.

LIPPMANN, R. P. An introduction to computing with neural nets. **ACM SIGARCH Computer Architecture News**, ACM New York, NY, USA, v. 16, n. 1, p. 7–25, 1988.

MASTERS, T. **Practical neural network recipes in C++**. [S.l.]: Morgan Kaufmann, 1993.

MICHELSEN, F. A.; FOSS, B. A. A dynamic model of the interaction between the chemical reactions and the residence time in a continuous digester. **Tappi journal**, v. 79, n. 4, p. 170–176, 1996.

NAVARRO, R. M. S. **Estudo dos diferentes tipos de processos de branqueamento de celulose objetivando a comparação entre seus metodos e a geração do potencial de poluentes em seus respectivos efluentes**. Tese (Doutorado) — [sn], 2004.

PADHIYAR, N.; GUPTA, A.; GAUTAM, A.; BHARTIYA, S.; III, F. J. D.; DASH, S.; GAIKWAD, S. Nonlinear inferential multi-rate control of kappa number at multiple locations in a continuous pulp digester. **Journal of Process Control**, Elsevier, v. 16, n. 10, p. 1037–1053, 2006.

PATTERSON, D. W. Artificial neural network theory. **Applications [MI. New Jersey: Prentice Hall**, 1996.

POLIT, M.; ESTABEN, M.; LABAT, P. A fuzzy model for an anaerobic digester, comparison with experimental results. **Engineering Applications of Artificial Intelligence**, Elsevier, v. 15, n. 5, p. 385–390, 2002.

POUGATCH, K.; SALCUDEAN, M.; GARTSHORE, I. A numerical model of the reacting multiphase flow in a pulp digester. **Applied mathematical modelling**, Elsevier, v. 30, n. 2, p. 209–230, 2006.

QIAN, Y.; LIU, H.; ZHANG, X.; TESSIER, P. J. Optimization of a wood chip refining process based on fuzzy relational models. **Computers & chemical engineering**, Elsevier, v. 21, p. S1137–S1142, 1997.

QUEIROZ, S. C. S.; GOMIDE, J. L.; COLODETTE, J. L.; OLIVEIRA, R. C. d. Influência da densidade básica da madeira na qualidade da polpa kraft de clones híbridos de eucalyptus grandis w. hill ex maiden x eucalyptus urophylla st blake. **Revista Árvore**, SciELO Brasil, v. 28, p. 901–909, 2004.

RAJESH, K.; RAY, A. Artificial neural network for solving paper industry problems: A review. CSIR, 2006.

RIBEIRO, R. N. **Utilização de redes neurais artificiais e tecnologia ft-nir para predição do número kappa em um processo kraft de cozimento de madeira em indústria de celulose**. Tese (Doutorado) — MSc thesis. Centro Universitário do Leste de Minas Gerais, Coronel . . . , 2007.

SHACKFORD, L. D.; SANTOS, C. A.; COLODETTE, J. L.; ALVES, E. F. Optimizing alkaline extraction for eucalyptus kraft pulp bleaching. **Tappi Journal**, TECH ASSOC PULP PAPER IND INC 15 TECHNOLOGY PARK SOUTH, NORCROSS, GA 30092 USA, v. 8, n. 1, p. 12–19, 2009.

SUSS, H.; NIMMERFROH, N. Hydrogen peroxide in chemical pulp bleaching. In: **ABTCP meeting on pulp bleaching at Vitoria, Espirito Santo, Brazil**. [S.l.: s.n.], 1996.

TIRYAKI, S.; HAMZAÇEBI, C. Predicting modulus of rupture (mor) and modulus of elasticity (moe) of heat treated woods by artificial neural networks. **Measurement**, Elsevier, v. 49, p. 266–274, 2014.

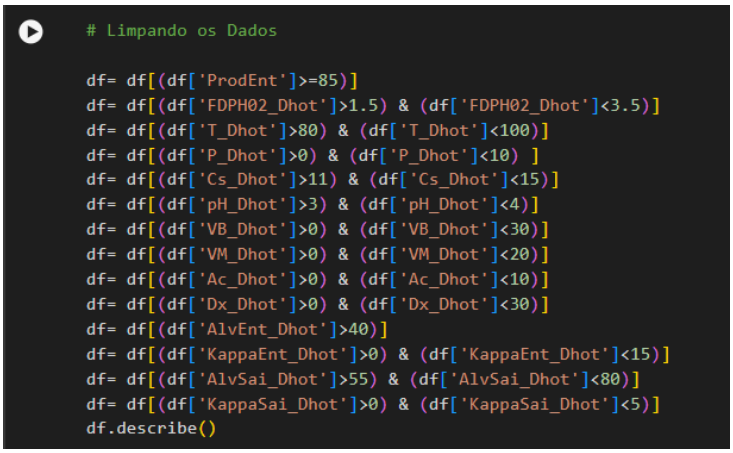
WISNEWSKI, P. A.; DOYLE, F. Model-based predictive control studies for a continuous pulp digester. **IEEE Transactions on Control Systems Technology**, IEEE, v. 9, n. 3, p. 435–444, 2001.

WISNEWSKI, P. A.; III, F. J. D. Control structure selection and model predictive control of the weyerhaeuser digester problem. **Journal of Process Control**, Elsevier, v. 8, n. 5-6, p. 487–495, 1998.

ZAINUDDIN, Z.; DAUD, W. R. W.; PAULINE, O.; SHAFIE, A. Wavelet neural networks applied to pulping of oil palm fronds. **Bioresource technology**, Elsevier, v. 102, n. 23, p. 10978–10986, 2011.

ANEXO A – Filtros utilizados nos dados de entrada para cada estágio

Figura 40 – Filtro utilizado nas variáveis de entrada do Estágio D_{hot}.

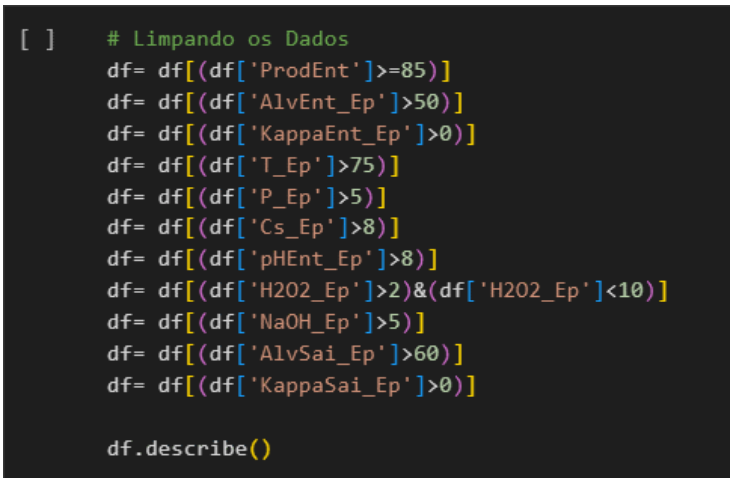


```
# Limpando os Dados

df= df[(df['ProdEnt']>=85)]
df= df[(df['FDPH02_Dhot']>1.5) & (df['FDPH02_Dhot']<3.5)]
df= df[(df['T_Dhot']>80) & (df['T_Dhot']<100)]
df= df[(df['P_Dhot']>0) & (df['P_Dhot']<10) ]
df= df[(df['Cs_Dhot']>11) & (df['Cs_Dhot']<15)]
df= df[(df['pH_Dhot']>3) & (df['pH_Dhot']<4)]
df= df[(df['VB_Dhot']>0) & (df['VB_Dhot']<30)]
df= df[(df['VM_Dhot']>0) & (df['VM_Dhot']<20)]
df= df[(df['Ac_Dhot']>0) & (df['Ac_Dhot']<10)]
df= df[(df['Dx_Dhot']>0) & (df['Dx_Dhot']<30)]
df= df[(df['AlvEnt_Dhot']>40)]
df= df[(df['KappaEnt_Dhot']>0) & (df['KappaEnt_Dhot']<15)]
df= df[(df['AlvSai_Dhot']>55) & (df['AlvSai_Dhot']<80)]
df= df[(df['KappaSai_Dhot']>0) & (df['KappaSai_Dhot']<5)]
df.describe()
```

Fonte: Próprio autor.

Figura 41 – Filtro utilizado nas variáveis de entrada do Estágio E_p.



```
[ ]  # Limpando os Dados
     df= df[(df['ProdEnt']>=85)]
     df= df[(df['AlvEnt_Ep']>50)]
     df= df[(df['KappaEnt_Ep']>0)]
     df= df[(df['T_Ep']>75)]
     df= df[(df['P_Ep']>5)]
     df= df[(df['Cs_Ep']>8)]
     df= df[(df['pHEnt_Ep']>8)]
     df= df[(df['H2O2_Ep']>2)&(df['H2O2_Ep']<10)]
     df= df[(df['NaOH_Ep']>5)]
     df= df[(df['AlvSai_Ep']>60)]
     df= df[(df['KappaSai_Ep']>0)]

     df.describe()
```

Fonte: Próprio autor.

Figura 42 – Filtro utilizado nas variáveis de entrada do Estágio D_1.

```
# Limpando os Dados
df= df[(df['ProdEnt']>=85)]
df= df[(df['AlvEnt_D1']>70)]
df= df[(df['AlvSai_D1']>80)]
df= df[(df['KappaEnt_D1']>3)]
df= df[(df['KappaSai_D1']>0)]
df= df[(df['T_D1']>80) & (df['T_D1']<90)]
df= df[(df['P_D1']>5)]
df= df[(df['Cs_D1']>10)]
df= df[(df['pH_D1']>2)]
df= df[(df['Ac_D1']>0)]
df= df[(df['Dx_D1']>0) & (df['Dx_D1']<15)]
df.describe()
```

Fonte: Próprio autor.

Figura 43 – Filtro utilizado nas variáveis de entrada do Estágio P.

```
# Limpando os Dados
df= df[(df['ProdEnt']>=85)]
df= df[(df['AlvEnt_Pf']>80)]
df= df[(df['KappaEnt_Pf']>0) & (df['KappaEnt_Pf']<3)]
df= df[(df['T_Pf']>70)  & (df['T_Pf']<90)]
df= df[(df['P_Pf']>0)]
df= df[(df['Cs_Pf']>5)]
df= df[(df['pH_Pf']>9)]
df= df[(df['VM_Pf']>0)]
df= df[(df['NaOH_Pf']>2) & (df['NaOH_Pf']<7)]
df= df[(df['H2O2_Pf']>2) & (df['H2O2_Pf']<7)]
df= df[(df['AlvSai_Pf']>85)]
df= df[(df['KappaSai_Pf']>0)]

df.describe()
```

Fonte: Próprio autor.

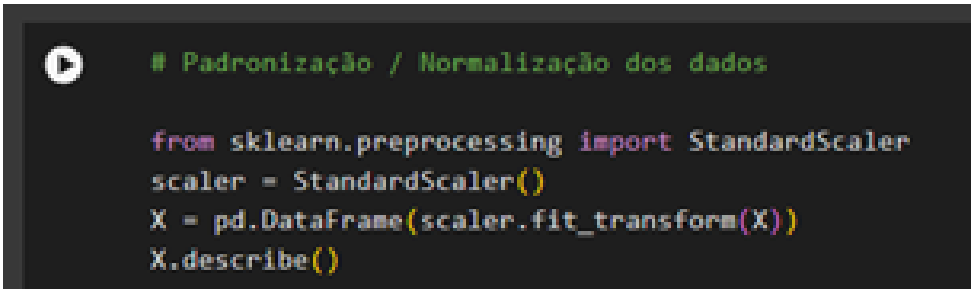
ANEXO B – Arquitetura das Redes Neurais Artificiais

Figura 44 – Código de carregamento do arquivo de dados em Excel.

```
#Endereço do Banco de Dados do Excel
df = pd.read_excel("Dhot_L2_P1.xlsx")
```

Fonte: Próprio autor.

Figura 45 – Código de padronização das variáveis de entrada.

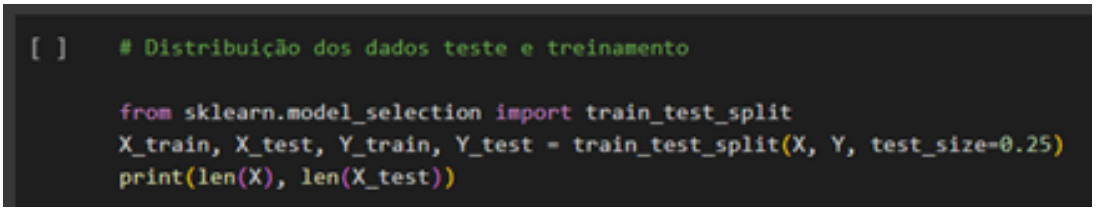


```
# Padronização / Normalização dos dados

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X = pd.DataFrame(scaler.fit_transform(X))
X.describe()
```

Fonte: Próprio autor.

Figura 46 – Código de criação dos dados de treinamento e teste.



```
[ ]   # Distribuição dos dados teste e treinamento

      from sklearn.model_selection import train_test_split
      X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25)
      print(len(X), len(X_test))
```

Fonte: Próprio autor.

Criação do modelo sequencial, que é uma estrutura na qual cada camada tem exatamente um tensor de entrada e um tensor de saída, com as camadas organizadas em uma pilha linear. A primeira camada é adicionada usando *layers. Dense*. Esta camada tem um número ajustável de unidades (neurônios), variando de 10 a 5000 com passos de 10, que é determinado dinamicamente pelo hiperparâmetro '*hp.Int*'. O tipo de função de ativação desta camada também é selecionado dinamicamente entre *ReLu*, *tanh* e função sigmoide, usando '*hp.Choice*'. A camada é configurada para receber entrada de tamanho 12 variáveis de entrada para o Estágio D_{hot} e 9 variáveis para os outros três Estágios (E_p, D_1 e P). Um loop é usado para adicionar de 0 a 3 camadas adicionais ou ocultas, dependendo do valor do hiperparâmetro. Cada uma dessas camadas adicionais também tem o número de unidades variando de 10 a 5000 com passos de 10 e o tipo de ativação determinados por hiperparâmetros ajustáveis. Uma camada de *Dropout* é potencialmente adicionada se '*hp. Boolean(dropout)*' for verdadeiro. A camada de *Dropout* ajuda a prevenir o *overfitting* durante o treinamento, descartando aleatoriamente uma proporção dos neurônios durante as atualizações de treinamento. Uma camada completamente conectada com uma única unidade e função de ativação Linear é adicionada ao final do modelo.

O modelo é compilado com um otimizador Adam, cuja taxa de aprendizado é escolhida entre três valores possíveis. A função de perda é configurada como erro quadrático médio MSE, e a métrica de desempenho é o erro absoluto médio MAE.

Figura 47 – Código de tunelamento do modelo da rede neural MLP empregado para todos os estágios de branqueamento utilizando o *Keras Tuner*.

```
[ ]   # Definir a Função do Modelo
      def build_model(hp):
          model = keras.Sequential()
      # Adiciona a primeira camada densa com o número de unidades ajustável
          model.add(layers.Dense(units=hp.Int("units", min_value=10, max_value=5000, step=10),
                                 activation=hp.Choice("activation", ["relu", "tanh", "sigmoid"]),input_shape=(12,)))
       # Adiciona camadas oculta com número de unidades e ativação ajustáveis
          for i in range(hp.Int('num_layers', min_value=0, max_value=3, step=1)):
              model.add(layers.Dense(units=hp.Int('units_' + str(i), min_value=10, max_value=5000, step=10),
                                     activation=hp.Choice("activation", ["relu", "tanh", "sigmoid"])))
          if hp.Boolean("dropout"):
              model.add(layers.Dropout(rate=0.25))
      # Adiciona a camada de saída com ativação linear
          model.add(layers.Dense(1, activation='linear'))
      # Compila o modelo com otimizador, função de perda e métricas
          model.compile(optimizer=keras.optimizers.Adam(hp.Choice('learning_rate', values=[1e-2, 1e-3, 1e-4])),
                        loss='mse', metrics=['mae'])
          return model
```

Fonte: Próprio autor.

B.1 ESTÁGIO DE DIOXIDAÇÃO A QUENTE

B.1.1 VARIÁVEL DEPENDENTE : *ALVURA*

O trecho de código mostrado na Figura 48 apresenta a arquitetura da rede neural sequencial criada com o uso da biblioteca Keras, que é uma interface de alto nível para a construção de redes neurais e que roda em cima de um *backend* como *Tensorflow*. O comando 'Sequential()' inicializa um modelo sequencial, que é um tipo de modelo do Keras que é composto por uma pilha linear de camadas, permitindo que uma camada tenha exatamente uma entrada tensor e uma saída tensor. A linha de comando 'Dense(4500, activation='tanh', input_shape=(12,))' na Figura 48, é a primeira camada da rede e é uma camada totalmente conectada, onde cada neurônio recebe entrada de todos os neurônios da camada anterior. Esta camada tem 4500 unidades ou neurônios e utiliza a função de ativação *tanh*. A função de ativação *tanh* é uma escolha comum e é uma função sigmoidal (ou seja, '*S-shaped*') que mapeia os valores de entrada para uma faixa entre -1 e 1. O 'input_shape' é definido como (12,), o que significa que a rede espera receber vetores de 12 variáveis como entrada, que corresponderia, por exemplo, a 12 *features* diferentes de um conjunto de dados. Segue-se outra camada densa com 3920 neurônios e a mesma função de ativação *tanh*. Não é necessário especificar o formato de entrada para as camadas subsequentes, pois o Keras faz essa inferência automaticamente a partir da camada anterior. Já a camada 'Dense(1, activation='linear')' é a camada final sendo também uma camada densa e tem uma única unidade. Ela utiliza uma função de ativação linear, o que significa que a saída da camada é a soma ponderada das entradas. Esta configuração é típica para problemas de regressão, onde se deseja prever um valor contínuo. O comando 'model.summary()' gera um resumo da arquitetura do modelo, incluindo o número de parâmetros treináveis e não treináveis em cada camada, bem como o total geral.

Figura 48 – Modelo da rede neural para predição da *ALVURA* - Estágio D_{hot}.

```
model = Sequential()
model.add(Dense(4500, activation='tanh', input_shape=(12,))),
model.add(Dense(3920, activation='tanh')),
model.add(Dense(1, activation='linear'))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 4500)              58500

 dense_1 (Dense)             (None, 3920)              17643920

 dense_2 (Dense)             (None, 1)                 3921

=================================================================
Total params: 17706341 (67.54 MB)
Trainable params: 17706341 (67.54 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Fonte: Próprio autor.

Na Figura 49 a linha de código ‘model.compile(optimizer=‘adam’, loss= ‘mse’, metrics=[‘mae’])’ é uma etapa crítica no processo de configuração de uma rede neural no Keras, onde é definido três elementos-chave necessários para a compilação do modelo. O Adam, que é uma escolha popular para muitos tipos de redes neurais, é um algoritmo de otimização baseado em gradiente estocástico que ajusta os pesos da rede durante o treinamento. Ele mantém uma taxa de aprendizagem para cada parâmetro do modelo e adapta essas taxas com base na estimativa do primeiro e segundo momentos dos gradientes. Isso ajuda a resolver dois problemas comuns em treinamento de redes neurais: minimizar a função de perda durante o treinamento e escolher uma taxa de aprendizagem apropriada. A Função MSE significa erro quadrático médio. Esta é a função de perda usada para avaliar o desempenho do modelo e no contexto de um problema de regressão, o MSE é calculado tomando a média do quadrado das diferenças entre os valores preditos pelo modelo e os valores reais. O objetivo do treinamento é minimizar essa função de perda ou seja, você quer que as previsões do modelo sejam o mais próximas possível dos valores reais. A Métrica MAE especifica as métricas que foram avaliadas pelo modelo durante o treinamento e teste. Neste caso, apenas o MAE é usado como métrica e é uma métrica comum para problemas de regressão, que calcula a média absoluta das diferenças entre os valores previstos e os valores reais. Diferente da função de perda, as métricas são usadas para monitorar o processo de treinamento e não são usadas para treinar o modelo. Em resumo, este comando configura o modelo para usar o otimizador Adam para ajustar os pesos, minimizar o erro quadrático médio durante o treinamento, e monitorar o erro médio absoluto como uma indicação de quão bem o modelo está performando.

Figura 49 – Parâmetros de compilação do modelo da rede neural para predição da *ALVURA* - Estágio D_{hot}.

```
[15]    model.compile(optimizer='adam', loss='mse', metrics=['mae'])
```

Fonte: Próprio autor.

O ‘model.fit()’ apresentado na Figura 50 é o método para treinar a rede neural, recebendo os dados de entrada ‘X_train’ e as etiquetas correspondentes ‘Y_train’, e realiza o processo de otimização de pesos através do *backpropagation* por um número definido de iterações, conhecidas como épocas que neste caso foi utilizado 50 épocas. Durante cada época, o algoritmo de otimização Adam neste caso, trabalha para minimizar a função MSE, ajustando os pesos com base no gradiente dessa função de perda em relação aos pesos. O ‘validation_data = X_train, Y_train’ é usado para fornecer um conjunto de dados de teste contra o qual o modelo é avaliado após cada época. Isso permite o monitoramento do desempenho do modelo em dados não vistos durante o treinamento, ajudando a detectar *overfitting*, onde o modelo se desempenha bem nos dados de treinamento mas mal nos dados de teste.

Figura 50 – Parâmetros do treinamento do modelo para predição da *ALVURA* - Estágio D_{hot}.

```
history = model.fit(X_train, Y_train, epochs=50, validation_data=(X_test, Y_test))
plt.plot(history.history['mae'])
plt.plot(history.history['val_mae'])
plt.title('Erro Médio Absoluto (MAE)')
plt.ylabel('Erro Médio Absoluto')
plt.xlabel('Epoch')
plt.legend(['treinamento', 'validação'], loc='upper right')
plt.show()
```

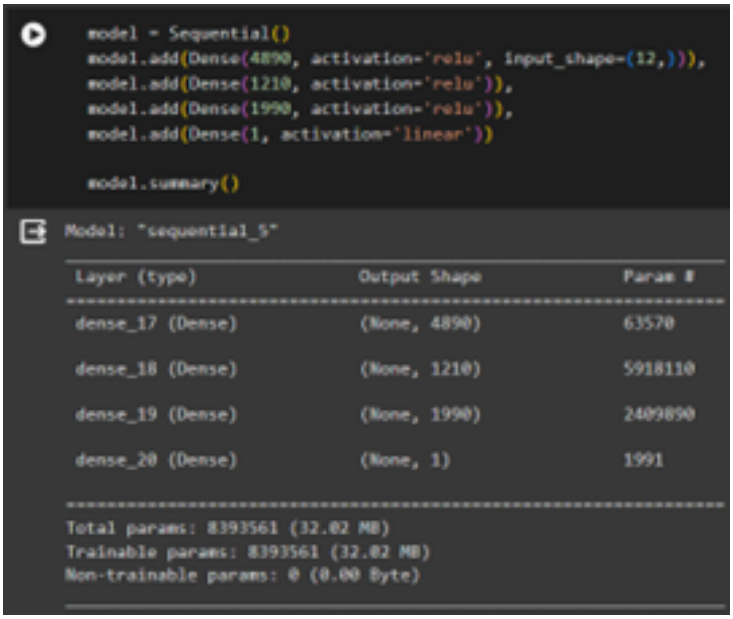
Fonte: Próprio autor.

As listas 'history.history['MAE']' e 'history.history['val_MAE']' registram a métrica de erro médio absoluto para cada época, tanto para os dados de treinamento quanto para os dados de teste ou validação, respectivamente. A chamada 'plt.plot()' da biblioteca *matplotlib* é usada para criar um gráfico de linha que visualiza estas métricas ao longo das épocas. A linha correspondente ao MAE refletindo o desempenho do modelo durante a fase de treinamento, enquanto a linha 'val_MAE' reflete o desempenho durante a validação. O propósito deste bloco de código é operacionalizar o treinamento do modelo enquanto se monitora e visualiza o desempenho do mesmo em termos de MAE, que é uma métrica chave na avaliação de modelos de regressão. A Figura 16 ajuda a identificar as tendências como o melhoramento ou degradação do desempenho do modelo ao longo das épocas.

B.1.2 VARIÁVEL DEPENDENTE: *NÚMERO KAPPA*

A arquitetura da rede neural sequencial foi construída com a biblioteca Keras no Python. O comando 'model = Sequential()' inicializa uma nova rede neural sequencial e é mostrado na Figura 51. Ele é adequado para uma pilha simples de camadas onde cada uma tem exatamente um tensor de entrada e um tensor de saída. A primeira camada 'model.add(Dense(4890, activation='relu', input_shape=(12,)))' adicionada ao modelo é uma camada totalmente conectada com 4890 neurônios. A função de ativação *ReLu* é usada aqui, que é uma escolha comum para as camadas ocultas por causa de sua eficiência e eficácia em não-linearidades. O 'input_shape=(12,)' indica que a entrada para a rede terá 12 características (variáveis independentes). Segue-se a primeira camada oculta completamente conectada com 1210 neurônios 'model.add(Dense(1210, activation='relu'))', também utilizando a função de ativação *ReLu*. Não é necessário especificar novamente o formato de entrada, pois o Keras automaticamente infere o tamanho de entrada desta camada do tamanho de saída da camada anterior. Na sequência, a segunda camada oculta completamente conectada desta vez com 1990 neurônios 'model.add(Dense (1990, activation='relu'))'. A função *ReLu* é mantida como função de ativação. Essa camada ajuda a aumentar a profundidade da rede, permitindo que ela aprenda padrões mais complexos nos dados. A última camada é uma camada completamente conectada com um único neurônio e função de ativação linear sendo representada por 'model.add(Dense(1, activation='linear'))'. Essa configuração é típica para uma tarefa de regressão onde o modelo prediz um valor contínuo como o número *KAPPA*.

Figura 51 – Modelo da rede neural para predição do número *KAPPA* - Estágio D_{hot}.



Fonte: Próprio autor.

A linha de código ‘model.compile(optimizer=‘adam’, loss=‘mse’, metrics=[‘mae’])’ mostrado na Figura 52 configura o processo de treinamento para um modelo de aprendizado de máquina no KERAS, uma biblioteca de aprendizado profundo em Python. O otimizador ‘optimizer=‘adam’ é um dos aspectos mais cruciais na configuração do treinamento de uma rede neural. Ele determina como os pesos sinápticos da rede são atualizados utilizando a função de custo ou de perda.

A função de perda ‘loss=‘mse”, mede a média dos quadrados das diferenças entre os valores preditos e os valores reais. Este é um critério comum para problemas de regressão, ajudando a quantificar o erro do modelo. Ao minimizar essa função durante o treinamento, o modelo aprende a reduzir a diferença entre a previsão e o valor real. As métricas são usadas para monitorar e avaliar o desempenho do modelo durante o treinamento e teste. A MAE ’metrics=[’mae’]’, é outra medida de erro para problemas de regressão, que fornece uma ideia da magnitude do erro ao calcular a média dos erros absolutos entre predições e valores reais. Ao contrário do MSE, o MAE fornece uma medida linear do erro, o que é útil para entender o desempenho em termos mais concretos e comparáveis.

Figura 52 – Parâmetros de compilação do modelo da rede neural para predição do número *KAPPA* - Estágio D_{hot}.

Fonte: Próprio autor.

O método ‘model.compile’ é chamado para configurar o processo de treinamento do modelo neural. O otimizador Adam é escolhido por sua eficácia em ajustar os pesos de maneira adaptativa, minimizando a função de custo, que neste caso é o erro quadrático médio (MSE). Este erro mede a discrepância quadrática entre as previsões do modelo e os valores reais, servindo como o critério de otimização principal. Além disso, o erro absoluto médio (MAE) é utilizado como uma métrica para avaliar a precisão das previsões durante o treinamento, fornecendo uma perspectiva clara sobre o desempenho geral do modelo em termos de erro médio. Essa linha de configuração é essencial para preparar o modelo para o treinamento, assegurando que todos os parâmetros necessários estejam definidos para guiar a otimização do modelo. Os comandos apresentados na Figura 53 de código Python, utilizando a biblioteca Keras para treinar um modelo de rede neural e a biblioteca *Matplotlib* para visualizar o progresso do treinamento, faz várias coisas importantes.

Na linha ‘history = model.fit(X_train, Y_train, epochs=50, validation_data=(X_test, Y_test))’ da Figura 17, o ‘model.fit’ é usado para treinar o modelo. É fornecido os dados de entrada ‘(X_train)’ e os rótulos/targets correspondentes ‘(Y_train)’. O argumento ‘epochs=50’ especifica que o modelo deve ser treinado por 50 épocas completas. Uma época é uma passagem completa pelos dados de treinamento. Já o parâmetro ’validation_data’ recebe dados de teste ou validação. Aqui, ’(X_test)’ e ’(Y_test)’ são usados para avaliar a performance do modelo após cada época, o que ajuda a monitorar problemas como *overfitting*.

Figura 53 – Parâmetros do treinamento do modelo para predição do número *KAPPA* - Estágio D_{hot}.

```
history = model.fit(X_train, Y_train, epochs=50, validation_data=(X_test, Y_test))
plt.plot(history.history['mae'])
plt.plot(history.history['val_mae'])
plt.title('Erro Médio Absoluto (MAE)')
plt.ylabel('Erro Médio Absoluto')
plt.xlabel('Epoch')
plt.legend(['treinamento', 'validação'], loc='upper right')
plt.show()
```

Fonte: Próprio autor.

As linhas ‘plt.plot(history.history[’mae’]’ e ‘plt.plot(history.history[’val_mae’])’ da Figura 53, plotam o erro médio absoluto MAE ao longo das épocas para os dados de treinamento e validação, respectivamente podendo ser acompanhado na Figura 17. O MAE é uma métrica de performance que mostra o erro médio entre as previsões do modelo e os valores reais. As funções ‘plt.title, plt.ylabel, plt.xlabel’ adicionam um título ao gráfico e rótulos aos eixos y e x, respectivamente e ‘plt.legend’ adiciona uma legenda ao gráfico, que ajuda a diferenciar entre as linhas de treinamento e validação.

A linha ‘plt.show()’ exibe o gráfico e é o comando que efetivamente faz com que a visualização apareça para o usuário. Este código é um exemplo típico de como treinar e monitorar um modelo de aprendizado de máquina. O uso de dados de validação durante o treinamento ajuda a evitar o *overfitting*, enquanto a visualização do MAE ao longo das épocas oferece *insights* valiosos sobre o desempenho do modelo, permitindo aos desenvolvedores ver como o modelo aprende ao longo do tempo e se ajustar conforme necessário. O gráfico resultante é essencial para uma avaliação visual rápida da convergência do modelo e da consistência do treinamento entre as diferentes fases do modelo.

B.2 ESTÁGIO DE EXTRAÇÃO COM PERÓXIDO DE HIDROGÊNIO

B.2.1 VARIÁVEL DEPENDENTE : *ALVURA*

O código apresentado na Figura 54 do Anexo B define a rede neural sequencial usando a biblioteca Keras, que é comumente utilizada para criar modelos de aprendizado profundo em Python. A seguir é mostrada uma análise detalhada do que cada parte do código está fazendo.

A linha inicia-se com o comando ’model = Sequential()’, que é um modelo linear de camadas. É chamado de ’sequencial’ porque cria o modelo camada por camada em sequência. A primeira camada ’model.add(Dense(3270, activation=tanh, input_shape=(9,)))’, é também conhecida como totalmente conectada, com 3270 neurônios (unidades). A função de ativação *tanh*, ou tangente hiperbólica, é utilizada, o que significa que a saída de cada neurônio será o valor da tangente hiperbólica da entrada. A tangente hiperbólica é uma função de ativação que varia de -1 a 1 e pode ajudar a regularizar a saída do neurônio. O ’input_shape=(9,)’ define o formato da camada de entrada, indicando que o modelo espera receber dados com 9 variáveis independentes. A camada oculta completamente conectada ‘model.add(Dense(2070, activation=’tanh’))’ adiciona outra camada densa com 2070 neurônios, também com a função de ativação *tanh*. A função de ativação é a mesma da primeira camada, sugerindo uma busca por consistência e simetria na transformação das características à medida que passam pela rede. E finaliza com uma camada completamente conectada de saída ‘model.add(Dense(1, activation=‘linear’))’, desta vez com um único neurônio. A função de ativação linear significa que a saída desta camada é uma combinação linear dos seus inputs. Esta configuração é típica para problemas de regressão, onde o modelo é usado para prever um valor numérico contínuo. Por último o comando ‘model.summary()’ que exibe um resumo do modelo, incluindo o número de camadas, a saída de cada camada e o número total de parâmetros que o modelo terá. Isso é útil para verificar a estrutura e a complexidade do modelo, e para garantir que tudo foi montado como esperado.

A arquitetura descrita pelo código indica um modelo com duas camadas bastante densas e uma camada de saída para regressão. O uso da função de ativação *tanh* nas camadas ocultas é uma escolha menos comum em comparação à *ReLu*, mas ainda assim válida e pode ser preferida em certos cenários. Um aspecto importante desse modelo é o seu tamanho, com um grande número de parâmetros devido ao alto número de neurônios, podendo haver uma preocupação com *overfitting*, porém há dados suficientes para treinar o modelo de maneira eficaz.

O método 'model.compile' é uma etapa essencial na construção de um modelo de aprendizado de máquina usando Keras, uma *API* de alto nível para redes neurais que roda sobre Software library for machine learning (*TensorFlow*). Este método configura o modelo para o treinamento, definindo vários parâmetros importantes, Figura 55 do Anexo B. No comando 'optimizer='Adam'' o Adam refere-se ao otimizador, que é um algoritmo de otimização baseado em gradiente estocástico. O Adam é conhecido por combinar os benefícios de dois outros extensões do gradiente estocástico: o método *Adaptive Gradient Algorithm* (AdaGrad) e o *Root Mean Square Propagation* (RMSProp). Por essa razão, o Adam é eficiente em termos de cálculo, requer pouca memória, é invariante à diagonalização do gradiente e é apropriado para problemas com grandes conjuntos de dados e/ou muitos parâmetros. A função perda 'loss='mse'' foi utilizada para medir a diferença entre os valores previstos pelo modelo e os valores verdadeiros. O objetivo do treinamento é minimizar essa função, o que ajudará o modelo a prever resultados tão próximos quanto possível dos reais. O MSE é particularmente popular para problemas de regressão. Além da função de perda, pode-se especificar métricas adicionais para avaliar o desempenho do seu modelo. O MAE significa erro absoluto médio, que mede a média das diferenças absolutas entre as previsões e as observações reais conforme o comando 'metrics=['mae']'. Diferente do MSE, o MAE não eleva as diferenças ao quadrado, o que significa que ele pode ser mais robusto a *outliers*. As métricas são usadas principalmente para monitoramento durante o treinamento e não afetam o processo de aprendizado do modelo (ou seja, não são usadas para otimização). Este código prepara o modelo para o treinamento, escolhendo algoritmos que são adequados para uma ampla gama de problemas e que são relativamente fáceis de configurar. O otimizador Adam é uma escolha padrão devido à sua eficiência e eficácia, enquanto o MSE e o MAE são métricas apropriadas para a avaliação de modelos de regressão, fornecendo informações úteis sobre o desempenho do modelo em termos da diferença entre as previsões do modelo e os valores reais. Ao compilar o modelo com esses parâmetros, o modelo é equipado com ferramentas para aprender a partir dos dados de maneira eficaz e para ter seu desempenho avaliado de maneira significativa. Este bloco de código da Figura 56 do Anexo B, realiza duas tarefas principais dentro do ciclo de desenvolvimento de um modelo de aprendizado de máquina: treina o modelo e, em seguida, visualiza o seu desempenho.

O Comando 'history = model.fit(X_train, Y_train, epochs=50, validation_data=(X_test, Y_test))' é usado para treinar o modelo. O conjunto de dados de treinamento '(X_train, Y_train)' e o modelo ajusta os parâmetros (pesos) para minimizar a função de perda que foi definida no 'model.compile'. Já o comando 'epochs=50' indica que o processo de treinamento deve passar 50 vezes pelo conjunto de dados inteiro (uma época é uma passagem completa pelos dados de treinamento). Além dos dados de treinamento, dados de teste ou validação '(X_test, Y_test)' são fornecidos. Após cada época, o modelo é avaliado em relação a este conjunto de dados para monitorar como está o desempenho em dados não vistos durante o treinamento. Isso ajuda a detectar *overfitting*, que é quando um modelo aprende a reconhecer padrões específicos dos dados de treinamento, mas não generaliza bem para novos dados. O resultado do treinamento é armazenado na variável '*history*'. Ele contém informações sobre a perda e as métricas de desempenho em cada época, tanto para treinamento quanto para validação.O comando 'plt.plot' plota o erro médio absoluto (MAE) para os dados de treinamento e validação em função do número da época. Já os comandos 'plt.title, plt.ylabel, plt.xlabel', são usados para adicionar um título ao gráfico e rótulos aos eixos Y e X, respectivamente. A legenda é adicionada ao gráfico para diferenciar entre as linhas que representam o treinamento e a validação atavés do comando 'plt.legend' e finalizando com o comando 'plt.show()' que exibe o gráfico. Isso permite visualizar como o erro muda ao longo do tempo durante o treinamento, conforme a Figura 22.

Figura 54 – Modelo da rede neural para predição da *ALVURA* - Estágio E_p.

```
model = Sequential()
model.add(Dense(3270, activation='tanh', input_shape=(9,))),
model.add(Dense(2070, activation='tanh')),
model.add(Dense(1, activation='linear'))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 3270)              32700

 dense_1 (Dense)             (None, 2070)              6770970

 dense_2 (Dense)             (None, 1)                 2071

=================================================================
Total params: 6805741 (25.96 MB)
Trainable params: 6805741 (25.96 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Fonte: Próprio autor.

Figura 55 – Parâmetros de compilação do modelo da rede neural para predição da *ALVURA* - Estágio E_p.

```
[15]   model.compile(optimizer='adam', loss='mse', metrics=['mae'])
```

Fonte: Próprio autor.

Figura 56 – Parâmetros do treinamento do modelo para predição da *ALVURA* - Estágio E_p.

```
history = model.fit(X_train, Y_train, epochs=50, validation_data=(X_test, Y_test))
plt.plot(history.history['mae'])
plt.plot(history.history['val_mae'])
plt.title('Erro Médio Absoluto (MAE)')
plt.ylabel('Erro Médio Absoluto')
plt.xlabel('Epoch')
plt.legend(['treinamento', 'validação'], loc='upper right')
plt.show()
```

Fonte: Próprio autor.

B.2.2 VARIÁVEL DEPENDENTE : *NÚMERO KAPPA*

A Figura 57 do Anexo B descreve a Próprio modelo da rede neural simples usando Keras, para a predição do *KAPPA* para o estágio E_p. Na sequência é mostrado o detalhamento e comentários sobre o código. A linha inicializa um modelo 'model = Sequential()' no Keras, que é uma pilha linear de camadas, onde cada camada tem exatamente um tensor de entrada e um tensor de saída. A primeira camada adicionada 'model.add(Dense(1010, activation='relu',input_shape= (9,)))', é configurada com 1010 neurônios e usa a função de ativação *ReLU (unidade linear retificada)*. O argumento indica que o vetor de recurso de entrada para o modelo terá 9 variáveis de entrada (variáveis independentes). Normalmente o modo dos dados de entrada é 'Densein-put_shape=(9,)'. A segunda camada é outra camada com 1010 neurônios 'model.add(Dense(1010, activation='relu'))', também usando a ativação *ReLu*. Essa camada recebeu entradas das 1010 saídas da camada anterior. A camada final ou de saída 'model.add(Dense(1, activation='linear'))' é uma camada com um único neurônio, usando uma função de ativação linear. Essa configuração é comum para tarefas de regressão que prevê um único valor contínuo do *KAPPA*. A linha 'model.summary()' imprimirá um resumo do modelo, mostrando a configuração de cada camada, incluindo o tipo de camada, a maneira de saída e o número de parâmetros. Isso foi útil para verificar a estrutura do modelo e entender como os dados fluem por ele. A linha de código fornecida 'model.compile(optimizer='adam', loss='mse', metrics= ['mae'])' descrito na Figura 58 do Anexo B, define as configurações de compilação do modelo de Rede Neural usando Keras. O parâmetro '*optimizer*' especifica o algoritmo de otimização a ser usado ao treinar o modelo. O otimizador Adam é especificado o que significa Estimativa de Momento Adaptativo. O Adam é um algoritmo de otimização popular no aprendizado profundo para lidar com gradientes esparsos em problemas ruidosos, ajustando automaticamente a taxa de aprendizagem durante o treinamento, o que o torna eficiente em termos de memória e computação.

O parâmetro 'loss-mse' define a função de perda a ser minimizada durante o treinamento. O MSE calcula a média dos quadrados dos erros, ou seja, a diferença quadrática média entre os valores estimados e o valor real. Esta função de perda é particularmente útil quando se tem valores numéricos contínuos para prever e o modelo deve penalizar erros maiores mais do que os menores, o que é típico em tarefas.

A métrica 'metrics=['mae']' é usada para especificar uma lista de métricas a serem avaliadas pelo modelo durante o treinamento e teste. Ao contrário do MSE, o MAE fornece um escore linear que faz a média das diferenças absolutas entre os valores previstos e os valores reais, dando uma ideia de quão erradas estão as previsões sem quadrar os erros.

Incluir o MAE como métrica permite monitorar outro aspecto do desempenho do modelo durante o treinamento, particularmente útil para entender a magnitude média dos erros de uma maneira mais interpretável do que o MSE. Ao compilar o modelo com essas configurações, configura o processo de treinamento para otimizar os pesos usando o otimizador Adam, visando minimizar o erro quadrático médio entre as previsões e os valores reais, além de monitorar o erro absoluto médio para medir o desempenho. Essa configuração é típica e eficaz para uma ampla gama de problemas de regressão.

O trecho de código fornecido e mostrado na Figura 59 do Anexo B demonstra como treinar um modelo de rede neural e visualizar seu histórico de treinamento, focando especificamente no Erro Absoluto Médio MAE durante o treinamento e validação. No comando 'history = model.fit(X_train, Y_train, epoch=50, validation_data = (X_test, Y_test))', o 'model.fit()' é utilizado para treinar o modelo com os dados de formação fornecidos '(X_train, Y_train)'. O parâmetro 'epochs=50' indica que o modelo deve ser treinado ao longo de 50 passagens completas do conjunto de dados de treinamento. O argumento 'validation_data' fornece dados sobre os quais avaliar a perda e quaisquer métricas de modelo no final de cada época. Aqui, ele foi utilizado para ver como o modelo se generaliza para novos dados '(X_test, Y_test)'. O dicionário 'history.history' contém os valores das métricas registradas no final de cada época durante o treinamento. Este trecho plota o MAE tanto para treinamento '(mae)' quanto para validação '(val_mae)'.

O comando 'plt.plot()' é usado para criar gráfico de linhas do MAE de treinamento e validação como mostrado na Figura 23. Diferentes linhas no gráfico ajudam a distinguir entre o desempenho no conjunto de treinamento e o conjunto de validação, mostrando como o erro de previsão do modelo diminui ao longo das épocas. O gráfico é intitulado e rotulado adequadamente, indicando que a métrica primária a ser visualizada é a MAE, que reflete a magnitude média dos erros nas previsões. A chamada 'plt.legend()' adiciona uma legenda no canto superior direito do gráfico, ajudando a identificar qual linha representa o MAE de treinamento e qual representa o MAE de validação. No geral, esse código treina a rede neural em uma tarefa de regressão e visualiza seu desempenho ao longo do tempo. Tais visualizações são cruciais para entender a dinâmica de aprendizagem do modelo, como a rapidez com que ele aprende, se está ocorrendo *overfitting* nos dados de treinamento e quando convergiu. O uso de dados de validação ajuda a garantir que o desempenho do modelo não seja apenas resultado da memorização dos dados de treinamento, mas generalizando bem para dados novos e inéditos.

Figura 57 – Modelo da rede neural para predição do número *KAPPA*- Estágio E_p.

```
[ ]   model = Sequential()
      model.add(Dense(1010, activation='relu', input_shape=(9,))),
      model.add(Dense(1010, activation='relu')),
      model.add(Dense(1, activation='linear'))

      model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 1010)              10100

 dense_4 (Dense)             (None, 1010)              1021110

 dense_5 (Dense)             (None, 1)                 1011

=================================================================
Total params: 1032221 (3.94 MB)
Trainable params: 1032221 (3.94 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Fonte: Próprio autor.

Figura 58 – Parâmetros de compilação do modelo da rede neural para predição do número *KAPPA* - Estágio E_p.

```
[15]   model.compile(optimizer='adam', loss='mse', metrics=['mae'])
```

Fonte: Próprio autor.

Figura 59 – Parâmetros do treinamento do modelo para predição do número *KAPPA* - Estágio E_p.

```
history = model.fit(X_train, Y_train, epochs=50, validation_data=(X_test, Y_test))
plt.plot(history.history['mae'])
plt.plot(history.history['val_mae'])
plt.title('Erro Médio Absoluto (MAE)')
plt.ylabel('Erro Médio Absoluto')
plt.xlabel('Epoch')
plt.legend(['treinamento', 'validação'], loc='upper right')
plt.show()
```

Fonte: Próprio autor.

B.3 ESTÁGIO DE DIOXIDAÇÃO

B.3.1 VARIÁVEL DEPENDENTE : *ALVURA*

O trecho de código fornecido na Figura 60 do Anexo B, detalha a configuração da rede neural usando Keras. O modelo é estruturado de maneira sequencial, indicando que cada camada alimenta a próxima sem ramificações ou saltos.

O comando ‘model = Sequencial()’ inicializa um modelo ’Sequencial’ que é uma pilha direta de camadas onde cada camada tem exatamente um tensor de entrada e um tensor de saída. É uma das maneiras mais simples de arquiteturas de redes neurais. A primeira camada ‘model.add(Dense(300, activation=‘tanh’, input_shape=(9,)))’ tem 300 neurônios e usa a função de ativação da tangente hiperbólica *tanh*. Ele espera tensores de entrada de modo ’(9,)’, o que implica que cada amostra de entrada tem 9 variáveis independentes. A função *tanh* produz valores entre -1 e 1 e é frequentemente usada em camadas ocultas devido ao seu manuseio eficaz de gradientes de desaparecimento em comparação com funções sigmoides em determinados cenários. As camadas subsequentes ‘model.add(Dense(1320,activation=‘tanh’))’ e ‘model.add(Dense(4580,activation=‘tanh’))’, expandem-se significativamente em tamanho, com 1320 e 4580 neurônios respectivamente, todos usando a função de ativação *tanh*. Esse aumento dramático na contagem de neurônios poderia potencialmente modelar relações muito complexas nos dados, mas também levanta preocupações sobre o *overfitting*, especialmente se a quantidade de dados de treinamento não for suficientemente grande. A camada final ‘model.add(Dense(1, activation=linear))’ é uma camada completamente conectada com um único neurônio, utilizando uma função de ativação linear. Essa configuração é típica para tarefas de regressão em que a saída é um único valor contínuo. A função ‘model.summary()’ imprimirá um resumo da arquitetura do modelo, fornecendo detalhes como o número de camadas, o número de neurônios em cada camada, as funções de ativação usadas e o número total de parâmetros treináveis.

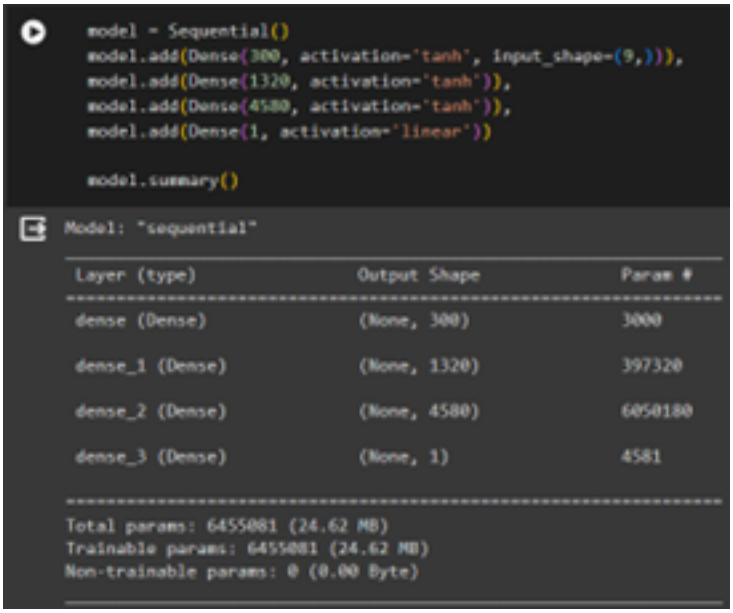
Isso é útil para uma visão geral rápida e para garantir que o modelo seja construído conforme o esperado. No geral, essa configuração de modelo usa camadas com um alto número de neurônios que podem potencialmente capturar padrões muito complexos nos dados. No entanto, o uso de grandes camadas também pode levar ao *overfitting*, fazendo com que o modelo tenha um bom desempenho em dados de treinamento, mas mal em dados de teste. O uso da função de ativação *tanh* em muitas camadas antes da saída pode afetar a velocidade de treinamento e convergência devido à natureza dos gradientes nas funções do *tanh*. Portanto, o ajuste cuidadoso dos parâmetros do modelo, as técnicas de regularização e a validação durante o treinamento foram essenciais para alcançar um bom desempenho. A linha de código ‘model.compile(optimizer=‘adam’, loss=‘mse’, metrics=[‘mae’])’ fornecida é crucial para configurar o modelo de treinamento em Keras, configurando seu processo de aprendizagem, Figura 61 do Anexo B. O otimizador ‘optimizer=‘adam’, especifica o método para atualizar o modelo com base nos dados que ele vê e sua função de perda. Adam é bem visto por sua eficiência em lidar com gradientes esparsos e sua capacidade de taxa de aprendizado adaptativo, tornando-o adequado para uma ampla gama de problemas. A função ‘loss=‘mse’ é o que o modelo está tentando minimizar. O MSE é comumente usado para problemas de regressão, é uma boa opção quando se tem dados numéricos contínuos e a magnitude do erro é importante para capturar, pois quadra os erros, amplificando assim erros maiores mais do que os menores.

As métricas são usadas para monitorar as etapas de treinamento e teste. Ao contrário da perda, as métricas não afetam diretamente o processo de treinamento, mas fornecem informações adicionais sobre o desempenho do modelo. O MAE, fornece um valor numérico fácil de interpretar que representa o quão longe as previsões estão em média. Ter o MAE como métrica é útil porque fornece uma medida mais intuitiva da magnitude do erro do que o MSE, pois não quadra os erros, tratando todos os erros linearmente. Ao compilar o modelo com essas configurações, ele é preparado para o treinamento com o otimizador Adam orientando a maneira como o modelo aprende, minimizando o erro quadrático médio para ajustar pesos e rastreando o erro absoluto médio para dar uma noção direta de quão precisas são as previsões do modelo em uma maneira mais compreensível. Essa configuração garante que o modelo seja otimizado e avaliado de acordo com padrões apropriados para tarefas de regressão. O trecho de código fornecido 'history = model.fit(X_train, Y_train, epochs=50, validation_data=(X_test, Y_test))' envolve o treinamento do modelo da Rede neural e a visualização do desempenho do modelo ao longo do tempo em termos de erro absoluto médio durante as fases de treinamento e validação, Figura 62 do anexo B. O comando 'model.fit()' é a função usada para treinar o modelo com os dados de treinamento fornecidos 'X_train' para entradas, 'Y_train' para metas'. Aqui, o modelo é treinado para 50 épocas, o que significa que o conjunto de dados será passado pela Rede Neural em 50 vezes.

O parâmetro 'validation_data' é dado como 'X_test, Y_test', permitindo que o modelo avalie seu desempenho em um conjunto de dados de teste separado após cada época. Isso ajuda a monitorar o quão bem o modelo generaliza para dados novos e invisíveis em comparação com os dados de treinamento, o que é crucial para evitar o *overfitting*. Após o treinamento, o MAE para as fases de treinamento e validação é plotado usando '*matplotlib*'. O MAE é uma métrica que mede a magnitude média dos erros em um conjunto de previsões, sem considerar sua direção (ou seja, agrega os valores absolutos dos erros).

O gráfico mostrado na Figura 28 fornece uma comparação visual do MAE a partir dos conjuntos de dados de treinamento e validação em todas as épocas. Essa visualização é crucial para identificar tendências, como melhorar o desempenho ou possíveis problemas como *overfitting* (onde o erro de treinamento diminui, mas o erro de validação não diminui ou até aumenta). No geral, esse trecho é uma parte essencial do desenvolvimento e da avaliação do modelo, fornecendo informações sobre os recursos de aprendizado e generalização do modelo ao longo do tempo. Ao monitorar o MAE, os desenvolvedores podem tomar decisões informadas sobre quando interromper o treinamento ou se ajustes precisam ser feitos na arquitetura do modelo ou no processo de treinamento para melhorar o desempenho ou reduzir o *overfitting*, porém ao observar a Figura 28 nota-se a inexistência de *overfitting*.

Figura 60 – Modelo da rede neural para predição da *ALVURA* - Estágio D_1.



```
model = Sequential()
model.add(Dense(300, activation='tanh', input_shape=(9,))),
model.add(Dense(1320, activation='tanh')),
model.add(Dense(4580, activation='tanh')),
model.add(Dense(1, activation='linear'))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 300)               3000

 dense_1 (Dense)             (None, 1320)              397320

 dense_2 (Dense)             (None, 4580)              6050180

 dense_3 (Dense)             (None, 1)                 4581

=================================================================
Total params: 6455081 (24.62 MB)
Trainable params: 6455081 (24.62 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Fonte: Próprio autor.

Figura 61 – Parâmetros de compilação do modelo da rede neural para predição da *ALVURA* - Estágio D_1.

```
[15]   model.compile(optimizer='adam', loss='mse', metrics=['mae'])
```

Fonte: Próprio autor.

Figura 62 – Parâmetros do treinamento do modelo para predição do número *ALVURA* - Estágio D_1.

```
history = model.fit(X_train, Y_train, epochs=50, validation_data=(X_test, Y_test))
plt.plot(history.history['mae'])
plt.plot(history.history['val_mae'])
plt.title('Erro Médio Absoluto (MAE)')
plt.ylabel('Erro Médio Absoluto')
plt.xlabel('Epoch')
plt.legend(['treinamento', 'validação'], loc='upper right')
plt.show()
```

Fonte: Próprio autor.

B.3.2 VARIÁVEL DEPENDENTE: *NÚMERO KAPPA*

O trecho de código na Figura 63 do Anexo B descreve a Próprio modelo da Rede Neural usando a biblioteca Keras em Python, que é uma escolha popular para a construção de modelos de aprendizado profundo. É utilizado o comando 'model = Sequential()' o que implica que as camadas são empilhadas linearmente. Cada camada tem um tensor de entrada e um tensor de saída. A primeira camada é uma camada densa (totalmente conectada) com 3930 neurônios 'model.add(Dense(3930, activation='relu', input_shape=(9,)))'. A função de ativação 'activation='relu'´ é utilizada e espera dados de entrada com o modo, o que significa que cada amostra tem 9 variáveis independentes. Esta camada produzirá vetores de 3930 dimensões.

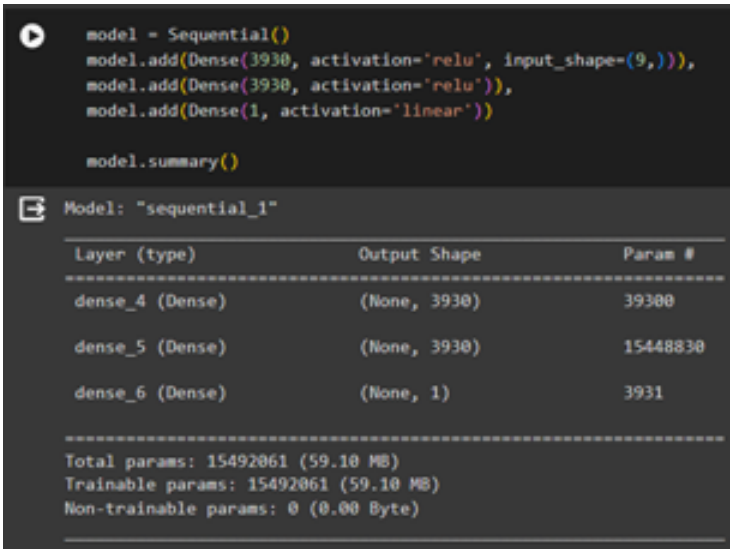
A segunda camada densa 'model.add(Dense(3930, activation='relu'))' também tem 3930 neurônios com ativação. Ele toma a saída da primeira camada densa como entrada *ReLu*. A camada final 'model.add(Dense(1, activation='linear'))' é outra camada densa com um único neurônio, usando uma função de ativação 'activation=linear'. Essa configuração garante que o modelo se destina a tarefas de regressão, pois gera um valor contínuo. Já a linha de comando 'model.summary()' imprime um resumo do modelo, mostrando a configuração de cada camada, incluindo o tipo de camada, maneira de saída, número de parâmetros, etc. Cada uma das duas camadas ocultas tem um número muito alto de neurônios (3930). Apesar que isso pode tornar o modelo bastante grande e potencialmente superajustado, porém há uma quantidade suficientemente grande de dados que minimiza este fato, mesmo aumentando o custo computacional. A maneira de entrada indica que o modelo leva 9 variáveis de entrada por amostra. A camada de saída de unidade única configurada com uma ativação sugere uma tarefa de regressão, conforme já mencionado anteriormente. No geral, esse código configura a Rede Neural profunda direta para um problema de regressão usando Keras, mas deve-se dedicar atenção especial com o tamanho do modelo em relação aos dados e recursos computacionais disponíveis. Esta linha de código 'model.compile(optimizer=adam, loss='mse', metrics=['mae'])', exemplificado na Figura 64 do Anexo B, configura o modelo para treinamento especificando o otimizador, a função de perda e as métricas para a predição do *KAPPA* no estágio D_1. O otimizador Adam especificado aqui é uma escolha popular para treinar modelos de aprendizado profundo e conhecido por sua eficácia no tratamento de gradientes esparsos em problemas ruidosos e é geralmente robusto em termos do tipo de problemas que pode tratar. A função de perda usada é MSE, que é uma escolha comum para problemas de regressão, pois calcula o quadrado da diferença entre os valores previstos e os valores reais, resultando em um número não negativo que representa o erro do modelo. Minimizar esse valor ajuda a refinar as previsões do modelo em relação aos valores reais. A métrica usada 'loss='mse" para avaliar o desempenho do modelo durante o treinamento e teste é MAE, que fornece uma pontuação linear que calcula a média das diferenças absolutas entre os valores previstos e os valores reais, proporcionando uma compreensão mais intuitiva da magnitude média do erro.

Os componentes escolhidos para a configuração do modelo, como o otimizador, a função de perda e a métrica, são ideais para um problema de regressão, alinhando-se com a arquitetura previamente definida. A utilização do MAE como métrica revelou-se eficaz durante o treinamento ao facilitar o monitoramento do desvio médio entre as previsões e os dados reais, oferecendo uma interpretação mais direta do que o erro quadrático medido pelo MSE. Embora o otimizador Adam seja geralmente eficiente em diversos contextos, foi crucial observar atentamente o treinamento para identificar possíveis sinais de *overfitting* ou *underfitting*. Isso envolveu ajustar parâmetros como a taxa de aprendizado e implementar técnicas de regularização. No entanto, conforme será demonstrado na Figura 29, esses problemas não foram observados durante o processo. Essa configuração torna o modelo para treinamento eficaz em tarefas de regressão, aproveitando os recursos de otimização de Adam e o MSE para minimização de perdas.

Este trecho de código Python, Figura 65 do Anexo B, demonstra como treinar um modelo de rede neural usando Keras para visualizar o desempenho em termos de erro médio absoluto MAE ao longo das épocas. A linha 'history=model.fit(X_train, Y_train, epochs=50, validation_data=(X_test, Y_test))' treina o modelo usando os dados de treinamento 'X_train e Y_train' ao longo de 50 épocas. Ele também avalia o modelo no conjunto de dados de validação 'X_test e Y_test' no final de cada época. O objeto 'history' captura as estatísticas de treinamento e validação em cada época, que inclui perdas e métricas especificadas durante a compilação do modelo. As linhas 'plt.plot(history.history['mae'])' e 'plt.plot(history.history['val_mae'])' representam o erro médio absoluto para o conjunto de treinamento '*mae*' e o conjunto de validação '*val_mae*' em todas as épocas. O uso de duas linhas separadas ajuda a comparar o desempenho do treinamento com o desempenho da validação, o que pode indicar quão bem o modelo generaliza para dados de teste.

As linhas 'plt.title(Erro Médio Absoluto (MAE))', 'plt.ylabel(Erro Médio Absoluto)' e 'plt.xlabel('epoch')' configuram o título do gráfico e os rótulos dos eixos y e x, respectivamente. Essas definições tornam o gráfico mais informativo ao explicitar a métrica representada e o parâmetro ao qual ela se relaciona. Adicionalmente, a linha 'plt.legend([trainamento, validação], loc=upper right)' insere uma legenda no gráfico, facilitando a distinção entre as curvas de treinamento e validação. Por fim, o comando 'plt.show()' é essencial, pois executa a exibição do gráfico. Esta função é crucial para materializar a visualização gráfica do MAE ao longo das épocas, permitindo uma análise direta do desempenho do modelo.

Figura 63 – Modelo da rede neural para predição do número *KAPPA* - Estágio D_1.



```
model = Sequential()
model.add(Dense(3930, activation='relu', input_shape=(9,))),
model.add(Dense(3930, activation='relu')),
model.add(Dense(1, activation='linear'))

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_4 (Dense)             (None, 3930)              39300

 dense_5 (Dense)             (None, 3930)              15448830

 dense_6 (Dense)             (None, 1)                 3931

=================================================================
Total params: 15492061 (59.10 MB)
Trainable params: 15492061 (59.10 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Fonte: Próprio autor.

Figura 64 – Parâmetros de compilação do modelo da rede neural para predição da *KAPPA* - Estágio D_1.

```
[15]  model.compile(optimizer='adam', loss='mse', metrics=['mae'])
```

Fonte: Próprio autor.

Figura 65 – Parâmetros do treinamento do modelo para predição do número *KAPPA* - Estágio D_1.

```
history = model.fit(X_train, Y_train, epochs=50, validation_data=(X_test, Y_test))
plt.plot(history.history['mae'])
plt.plot(history.history['val_mae'])
plt.title('Erro Médio Absoluto (MAE)')
plt.ylabel('Erro Médio Absoluto')
plt.xlabel('Epoch')
plt.legend(['treinamento', 'validação'], loc='upper right')
plt.show()
```

Fonte: Próprio autor.

B.4 ESTÁGIO DE PEROXIDAÇÃO

B.4.1 VARIÁVEL DEPENDENTE: *ALVURA*

Conforme ilustrado na Figura 66 do Anexo B, o trecho de código define a rede neural simples utilizando a biblioteca Keras em Python, especificamente estruturada para tarefas de regressão, conforme evidenciado pela ativação linear da camada final. O comando 'model = Sequential()' inicializa um modelo *Sequential*, que representa um dos tipos mais fundamentais de modelos de aprendizado profundo em Keras. A natureza sequencial implica que cada camada processa um tensor de entrada e produz um tensor de saída de maneira ordenada. A primeira camada, 'model.add(Dense(224, activation='tanh', input_shape=(9,)))', é densa e totalmente conectada com 224 neurônios, utilizando a função de ativação *tanh* (tangente hiperbólica). Esta função é escolhida por produzir valores no intervalo de -1 a 1 e por seu gradiente suave, o que ajuda a evitar oscilações abruptas nos valores de saída, minimizando a instabilidade numérica. Ela espera dados de entrada com nove variáveis, como especificado por 'input_shape=(9,)'. A camada de saída, 'model.add(Dense(1, activation='linear'))', consiste em uma única unidade com ativação linear, adequada para prever valores contínuos. A ativação linear permite que o valor de saída seja uma combinação linear direta das entradas e dos pesos, típica em contextos de regressão. O comando 'model.summary()' produz um resumo detalhado da configuração do modelo, incluindo o número de parâmetros e suas conexões, facilitando a verificação e a validação da arquitetura proposta. Este modelo, embora simples com apenas duas camadas, é apto para resolver problemas menos complexos, mas pode não ser suficiente para conjuntos de dados mais intrincados que requerem arquiteturas mais profundas. A escolha da função *tanh* na primeira camada é menos comum em aprendizado profundo contemporâneo, onde a função *ReLu* e suas variantes predominam devido à eficácia no tratamento de gradientes que desaparecem. No entanto, a *tanh* foi mais vantajosa em função das características específicas dos dados. A arquitetura do modelo, com sua saída única e ativação linear, confirma sua adequação para tarefas de regressão, ideal para prever variáveis contínuas como encontradas em processos industriais. Globalmente, esta configuração serve como um exemplo claro de Rede Neural para regressão, sendo particularmente útil para propósitos experimentos iniciais em um novo conjunto de dados. O otimizador Adam é amplamente utilizado para treinar modelos de aprendizado profundo, reconhecido por sua eficiência no manejo de gradientes esparsos em contextos ruidosos. É notável por sua robustez diante de diversos tipos de problemas. Ele ajusta a taxa de aprendizado dinamicamente, facilitando uma convergência mais rápida e eficiente, como ilustrado na Figura 67 do Anexo B. A função de perda 'loss='mse'', que representa o Erro Quadrático Médio (MSE), é comumente empregada em problemas de regressão. Ela calcula o quadrado das diferenças entre os valores previstos e os reais, dando maior ênfase aos erros mais significativos, o que é especialmente vantajoso em regressões para direcionar o modelo a previsões mais acuradas. A métrica 'metrics=['mae']', que utiliza o Erro Médio Absoluto (MAE), é adotada para avaliar o desempenho do modelo durante o treinamento e testes.

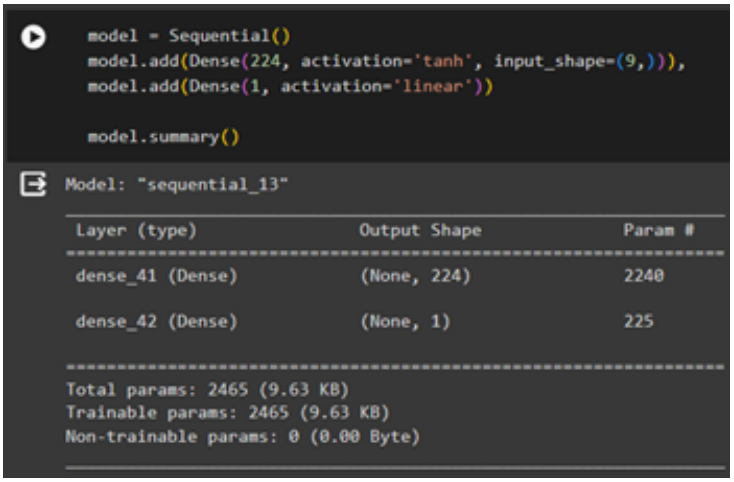
Diferentemente do MSE, o MAE fornece uma pontuação linear, computando a média das diferenças absolutas entre os valores previstos e os reais. Esta métrica é frequentemente mais intuitiva que o MSE, pois expressa o erro médio nas mesmas unidades da variável de saída. A combinação do otimizador Adam com o MSE constitui uma abordagem robusta para uma variedade de tarefas de predição. A capacidade adaptativa de Adam em ajustar a taxa de aprendizado o torna adequado para conjuntos de dados com escalas variadas de características. Incluir o MAE como métrica proporciona uma visão adicional sobre o desempenho do modelo, oferecendo *insights* sobre a magnitude média do erro, o que facilita a interpretação para quem avalia os resultados do modelo. Monitorar tanto o MSE quanto o MAE durante o treinamento pode revelar uma compreensão mais completa do desempenho do modelo e indicar áreas de potencial melhoria, particularmente em relação à magnitude dos erros. Em suma, esta configuração prepara o modelo para um treinamento eficaz e uma avaliação de desempenho aprofundada, maximizando os recursos de otimização do Adam e o *feedback* detalhado oferecido pelas métricas MSE e MAE. Tal configuração é ideal para tarefas de regressão que exigem previsões precisas e acuradas. O trecho de código Python apresentado na Figura 68 do Abexo B foi desenvolvido para treinar a Rede Neural e visualizar seu desempenho ao longo de várias épocas de treinamento, com um enfoque particular no Erro Médio Absoluto (MAE). Abaixo, uma análise detalhada dos principais componentes e suas funcionalidades. A linha 'history = model.fit(X_train,Y_train,epochs=50,validation_data=(X_test, Y_test))' inicia o treinamento do modelo com os conjuntos de dados de treinamento '(X_train, Y_train)' por 50 épocas, simultaneamente avaliando-o em um conjunto de dados de validação '(X_test, Y_test)'.

O objeto '*history*' registra as estatísticas de treinamento e validação após cada época, incluindo perdas e métricas adicionais especificadas durante a compilação do modelo, neste caso, o MAE. As linhas 'plt.plot(history.history['mae']) e plt.plot(history.history['val_mae'])' traçam o MAE para os conjuntos de dados de treinamento e validação ao longo das 50 épocas.

Os acessos 'history.history['mae'] e history.history['val_mae']' permitem visualizar o desempenho do modelo em ambos os conjuntos de dados ao longo do tempo, sendo plotados como linhas separadas para facilitar a comparação visual. Os comandos 'plt.title(Erro Médio Absoluto (MAE))', 'plt.ylabel(Erro Médio Absoluto) e plt.xlabel(Epoch)' configuram o título do gráfico e os rótulos dos eixos y e x, respectivamente. O título e os rótulos especificam claramente que o gráfico detalha o MAE, e o eixo x indica o número de épocas. O comando 'plt.legend([treinamento, validação], loc= upper left)' adiciona uma legenda ao gráfico, diferenciando as linhas de MAE de treinamento e validação. A função 'plt.show()' é crucial pois renderiza o gráfico na tela. Esta é a etapa final que agrega todos os comandos de plotagem para exibir a análise visual do MAE do modelo durante o treinamento e validação. Ao analisar o MAE para os conjuntos de treinamento e validação, conforme ilustrado na Figura 34, não foram observados indícios de *overfitting* ou *underfitting*.

A diminuição do MAE ao longo das épocas sugere melhorias contínuas no desempenho do modelo. Por outro lado, a observação de um platô ou um aumento no MAE pode indicar desafios como uma taxa de aprendizagem inadequada, complexidade insuficiente do modelo, ou a necessidade de expansão do conjunto de dados de treinamento. A visualização dos resultados desempenha um papel crucial na otimização do modelo, oferecendo um meio claro e intuitivo de avaliar a capacidade de aprendizado do modelo e sua generalização para novos dados. Esta abordagem integra efetivamente o monitoramento do processo de treinamento com a análise de desempenho, constituindo uma ferramenta valiosa para o desenvolvimento e aperfeiçoamento do modelo. Este gráfico utiliza uma linguagem mais formal e detalhada, adequada para um contexto acadêmico, e enfatiza a importância de monitorar o desempenho do modelo para evitar problemas comuns e ajustar adequadamente os parâmetros do modelo.

Figura 66 – Modelo da rede neural para predição do número *ALVURA* - Estágio *P*.



Fonte: Próprio autor.

Figura 67 – Parâmetros de compilação do modelo da rede neural para predição da *ALVURA* - Estágio *P*.

Fonte: Próprio autor.

Figura 68 – Parâmetros do treinamento do modelo para predição do número *ALVURA* - Estágio *P*.

Fonte: Próprio autor.

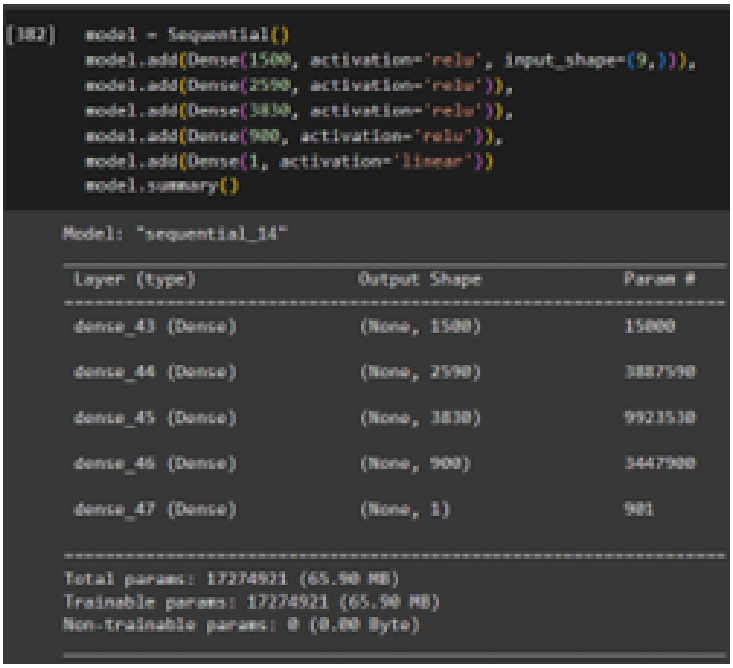
B.4.2 VARIÁVEL DEPENDENTE: *NÚMERO KAPPA*

O trecho de código apresentado na Figura 69 do Anexo B, ilustra a construção da Rede Neural utilizando a biblioteca Keras em Python, com uma arquitetura de modelo sequencial. Inicialmente, o modelo é configurado como um objeto 'Sequential()', caracterizando-se como uma pilha linear de camadas. Este método é comumente adotado em Keras quando cada camada é projetada para processar um único tensor de entrada e produzir um tensor de saída. A primeira camada completamente conectada, 'model.add(Dense(1500, activation='relu', input_shape=(9,)))', dispõe de 1500 neurônios e utiliza a função de ativação *ReLu*. Esta camada é especificada para receber entradas com 9 variáveis independentes, representando a maneira do tensor de entrada. Segue-se a segunda camada completamente conectada, 'model.add(Dense(2590, activation='relu'))', que contém 2590 neurônios, igualmente com ativação *ReLu*. A terceira camada completamente conectada, 'model.add(Dense(3830, activation='relu'))', compreende 3830 neurônios com a mesma ativação. Já a quarta camada completamente conectada, 'model.add(Dense(900, activation='relu'))', possui 900 neurônios, também com ativação *ReLu*. A camada de saída, 'model.add(Dense(1, activation='linear'))', consiste em um único neurônio com função de ativação linear, ideal para tarefas de regressão onde o objetivo é prever um valor contínuo. O comando 'model.summary()' é executado para gerar um resumo do modelo, destacando o número de camadas, seus tipos, modos de saída e o total de parâmetros, tanto treináveis quanto não treináveis. O modelo utiliza um alto número de neurônios nas camadas ocultas, o que potencialmente permite aprender padrões complexos nos dados. Contudo, isso também pode predispor o modelo ao *overfitting*, especialmente na ausência de uma quantidade adequada de dados ou estratégias eficazes de regularização. A escolha da ativação *ReLu* contribui para mitigar o problema do desvanecimento do gradiente durante o treinamento, facilitando o aprendizado em redes profundas. Esta arquitetura demanda considerável capacidade computacional devido ao grande número de neurônios por camada, indicando sua aplicabilidade a tarefas regressivas complexas com dados suficientes para treinamento efetivo. Na Figura 70 do Anexo B, a linha 'model.compile(optimizer=adam, loss=mse, metrics=[mae])' em uma configuração de modelo Keras é essencial para o preparo do modelo para o treinamento. O parâmetro 'optimizer=adam' designa o uso do otimizador Adam, uma extensão do método de descida de gradiente estocástico amplamente utilizada em aprendizado profundo. Este otimizador é reconhecido por sua eficiência em lidar com gradientes esparsos e objetivos não estacionários, elementos típicos no treinamento de redes neurais. A opção 'loss=mse' estabelece a função de perda como Erro Quadrático Médio (MSE), uma escolha comum para modelos de regressão. O MSE calcula a média dos quadrados das diferenças entre valores previstos e reais, sendo particularmente eficaz para penalizar erros maiores e impulsionar o modelo a focar nos pontos de dados mais desafiadores, o que pode resultar em previsões mais acuradas. Ao especificar 'metrics=[mae]', o modelo passará a monitorar o Erro Absoluto Médio (MAE) durante o treinamento e a avaliação. O MAE quantifica a magnitude média dos erros de previsões, sem diferenciar a direção dos erros, tratando-os como valores positivos.

Essa métrica oferece uma medida direta e compreensível da média dos erros. Utilizar o Adam permite que o modelo se beneficie de ajustes adaptativos na taxa de aprendizado, o que pode acelerar e tornar mais eficaz a convergência em relação à descida de gradiente tradicional. Com a função de perda 'MSE', o modelo é otimizado de maneira a minimizar significativamente os erros, adequando-se especialmente a conjuntos de dados com alta variabilidade. Esse método de otimização proporciona uma direção clara para a redução de grandes discrepâncias. Incluir o MAE como métrica fornece uma visão intuitiva e facilmente interpretável do desempenho do modelo, fazendo dela uma opção vantajosa para apresentação de resultados a *stakeholders* que não estão familiarizados com interpretações baseadas em valores ao quadrado. Essa configuração de compilação é ideal para o treinamento de modelos de regressão robustos, combinando um otimizador eficiente com uma função de perda e métricas que se alinham aos objetivos de minimizar erros e aprimorar a precisão das previsões. O trecho de código apresentado na Figura 71 do Anexo B está associado ao treinamento do modelo de *machine learning* utilizando a biblioteca Keras e à visualização do desempenho do modelo em termos de Erro Médio Absoluto (MAE). O comando 'history = model.fit(X_train, Y_train, epochs=50, validation_data=(X_test, Y_test))' inicia o treinamento do modelo com os conjuntos de dados 'X_train' e 'Y_train', especificando um total de 50 épocas. Durante este processo, os conjuntos 'X_test' e 'Y_test' são empregados como dados de validação para monitorar o desempenho do modelo em dados não vistos anteriormente. Os comandos 'plt.plot(history.history[mae])' e 'plt.plot(history.history[val_mae])' são utilizados para plotar o histórico de Erro Médio Absoluto nos conjuntos de treinamento 'mae' e de validação 'val_mae', respectivamente. Esses gráficos facilitam a visualização de como o erro do modelo evolui ao longo das épocas, tanto no treinamento quanto na validação. As configurações do gráfico incluem a função 'plt.title(Erro Médio Absoluto (MAE))', que define o título do gráfico. Os comandos 'plt.ylabel(Erro Médio Absoluto)' e 'plt.xlabel('Epoch')' estabelecem os rótulos dos eixos Y e X, respectivamente.

Adicionalmente, 'plt.legend([treinamento, validação], loc=upper left)' adiciona uma legenda ao gráfico, indicando quais linhas representam os dados de treinamento e de validação, posicionada no canto superior direito. O comando 'plt.show()' é utilizado para exibir o gráfico configurado, oferecendo uma ferramenta visual crucial para analisar a convergência do modelo e identificar possíveis problemas como *overfitting* ou *underfitting*, evidenciados pela divergência entre as curvas de treinamento e validação. Este tipo de visualização é fundamental para ajustar os parâmetros do modelo e entender seu comportamento ao longo das diversas fases do treinamento.

Figura 69 – Modelo da rede neural para predição do número *KAPPA* - Estágio *P*.

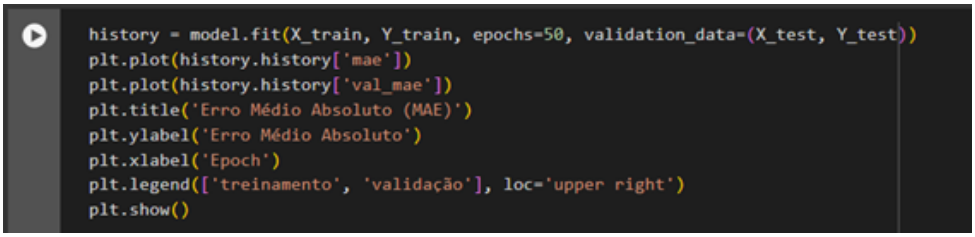


Fonte: Próprio autor.

Figura 70 – Parâmetros de compilação do modelo da rede neural para predição da *KAPPA* - Estágio *P*.

Fonte: Próprio autor.

Figura 71 – Parâmetros do treinamento do modelo para predição do número *KAPPA* - Estágio *P*.



Fonte: Próprio autor.