PEDRO OLIVEIRA ANNONI FARAH

MODELAGEM DE SENSOR VIRTUAL PARA ESTIMATIVA DA QUANTIDADE DE VAPOR GERADA EM UMA CALDEIRA INDUSTRIAL UTILIZANDO REDES NEURAIS ARTIFICIAIS

LAVRAS - MG

2022

PEDRO OLIVEIRA ANNONI FARAH

MODELAGEM DE SENSOR VIRTUAL PARA ESTIMATIVA DA QUANTIDADE DE VAPOR GERADA EM UMA CALDEIRA INDUSTRIAL UTILIZANDO REDES NEURAIS ARTIFICIAIS

Monografia apresentada à Universidade Federal de Lavras, como parte das exigências do Curso de Engenharia de Controle e Automação, para a obtenção do título de Bacharel.

Prof. Dr. Wilian Soares Lacerda

Orientador

LAVRAS - MG

2022

PEDRO OLIVEIRA ANNONI FARAH

MODELAGEM DE SENSOR VIRTUAL PARA ESTIMATIVA DA QUANTIDADE DE VAPOR GERADA EM UMA CALDEIRA INDUSTRIAL UTILIZANDO REDES NEURAIS ARTIFICIAIS

MODELING A VIRTUAL SENSOR TO ESTIMATE THE QUANTITY OF STEAM GENERATED IN AN INDUSTRIAL BOILER USING ARTIFICIAL NEURAL NETWORKS

Monografia apresentada à Universidade Federal de Lavras, como parte das exigências do Curso de Engenharia de Controle e Automação, para a obtenção do título de Bacharel.

APROVADA em 12 de dezembro de 2022

Dr. Fábio Domingues de Jesus	UFLA
Me. Franck Morais de Oliveira	UFLA
Dr. Wilian Soares Lacerda	UFLA

Prof. Dr. Wilian Soares Lacerda

Orientador

LAVRAS - MG

2022

Dedico ao meu pai Ricardo e à minha mãe Fabiana por tudo que fizeram e fazem por mim.

AGRADECIMENTOS

Agradeço em primeiro lugar a Deus, pelo dom da vida e que sempre me deu forças e me capacitou em todos os momentos difíceis.

Agradeço aos meus pais e minhas irmãs que sempre me apoiaram em todos os momentos.

Agradeço ao professor Wilian Soares Lacerda que me orientou neste projeto e sempre me auxiliou com paciência e atenção.

Agradecimentos também a José Geraldo da Silva Moreira, Fernanda Drumond, Dayane Machado e Luciano Melo que, de uma forma ou de outra, auxiliaram no desenvolvimento deste trabalho.

Resumo

Em muitas aplicações industriais, a geração e o consumo de vapor são de extrema importância para o funcionamento dos processos produtivos. Sua geração muitas vezes está ligada à utilização de caldeiras industriais alimentadas por diferentes combustíveis, onde a reação de combustão fornece a energia necessária para a vaporização da água, que geralmente se encontra ao redor dos tubos onde o processo acontece. Acompanhar e analisar a taxa produzida de vapor que está sendo gerada e distribuída para as subáreas de uma siderúrgica é de grande valor e depende muitas vezes da implantação de um sensor físico responsável por este monitoramento. No entanto, estes sensores são sujeitos a falhas, impossibilidade de instalação ou compra, erros de medida, atraso, entre outros fatores. Este trabalho propõe o desenvolvimento de um sensor virtual através da utilização de Redes Neurais Artificiais para estimar a taxa de vapor produzida por uma caldeira industrial a partir das taxas de combustível que a alimentam. Foi utilizado a rede tipo *Multilayer Perceptron* para a criação de um modelo regressor que alcançou resultados positivos tendo como métricas de avaliação o Erro Médio Absoluto alcançando um valor igual a 2,13, Erro Médio Absoluto Percentual igual a 0,0263 (ou 2,63%) e Coeficiente de Determinação igual a 0,813 e que foi implantado em um ambiente de simulação e visualização dos dados em tempo real.

Palavras-chave: Sensor virtual. Redes Neurais Artificiais. Caldeiras Industriais.

Abstract

In many industrial applications, the generation and consumption of steam is extremely important for the functioning of production processes. Its generation is often linked to the use of industrial boilers powered by different fuels, where the combustion reaction in contact with water in its liquid state provides the energy needed for vaporization. Monitoring and analyzing the rate of steam produced and distributed to the subareas of a steel mill is of great value and often depends on the implementation of a physical sensor responsible for this monitoring. However, these sensors are subject to failure, impossibility of installation or purchase, measurement errors, delay, among other factors. This work proposes the development of a soft sensor through the use of Artificial Neural Networks to estimate the rate of steam produced by an industrial boiler from the rates of fuel that feed it. The training of the Multilayer Perceptron algorithm was used to create a regressor model that achieved positive results, taking as evaluation measurements the Mean Absolute Error reaching a value equal to 2,13, Mean Absolute Percentage Error equal to 0,0263 (or 2,63%) and the Coefficient of Determination equal to 0,813, and was implemented in a real-time simulation and data visualization environment.

Keywords: Soft Sensor. Artificial Neural Networks. Industrial Boilers.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

GLP Gás Liquefeito de Petróleo

LDG *Linz-Donawitz Gas*

BFG *Blast Furnace Gas*

COG *Coke Oven Gas*

PCA Análise dos Principais Componentes

ReLU *Rectified Linear Unit*

MSE *Mean Squared Error*

MAE *Mean Absolute Error*

MAPE *Mean Absolute Percentage Error*

GPU *Graphics Processing Unit*

API *Application Programming Interface*

PIMS *Plant Information Management System*

OPC *Open Platform Communication*

HDA *Historical Data Access*

SUMÁRIO

1 Introdução

Os sensores virtuais constituem uma ferramenta de grande utilidade para diversas aplicações onde é necessário que haja a substituição de um sensor físico. Muitas vezes isto é necessário pois a medição do sensor físico é de tal modo crítica que, caso houvesse falhas em seu funcionamento que levasse a perda de dados, haveria um grande comprometimento do funcionamento do sistema que monitora. Outro motivo pelo qual se propõe a utilização de sensores virtuais é a possibilidade de se comparar o valor físico real com o estimado. Sensores físicos podem, por mau funcionamento e descalibração, passar a fornecer valores de medição que não condizem com a realidade, o que em muitos casos pode ser ainda mais danoso do que a parada em seu funcionamento.

A utilização das Redes Neurais, por sua vez, é apresentada como uma maneira de se desenvolver esse sensor virtual, a partir de uma base de dados histórica que de algum modo é processada, a fim de se obter informação útil. Sabe-se o grande poder proporcionado por esses algoritmos que, entre várias outras funcionalidades, permitem que dados de saída sejam inferidos a partir de dados de entrada e do mapeamento entre ambos que acontece durante o seu treinamento.

1.1 Motivação

Em diversos contextos surgem aplicações onde se permite a união dos sensores virtuais com as Redes Neurais Artificiais, a fim de possibilitar variados ganhos a partir da substituição de um sensor físico. Este trabalho propõe a utilização das Redes Neurais como unidade formadora de um sensor virtual aplicado a um caso real de uma indústria siderúrgica.

Sabe-se que a utilização de vapor em processos industriais é extremamente difundida nas plantas e aparece tanto sob o pretexto de consumo como de geração. No caso aqui apresentado, o vapor é gerado a partir de um sistema constituído por 4 caldeiras industriais instaladas na central termoelétrica da planta. Através da reação de combustão presente no interior dessas caldeiras, que acontece devido ao contato de diversas fontes combustíveis com o ar, há a liberação de calor que aquece a água no estado líquido até que haja a sua vaporização. O vapor quente de alta pressão que é gerado, é utilizado por diversos outros processos de produção que necessitam que haja uma redução da pressão. Apesar de a sua principal utilidade ser fornecer energia térmica para os outros processos dependentes, essa energia também pode ser utilizada para a produção de energia elétrica. Isso mostra que existe

toda uma rede de equipamentos e processos produtivos para os quais esta geração de vapor é essencial.

Desse modo, é fundamental que se tenha acesso à quantidade de vapor gerada a partir das quantidades de combustíveis que alimentam a caldeira. Nesse sistema termoelétrico em questão, existem medidores instalados que fornecem essas informações, funcionando como sensores físicos que enviam e armazenam os dados coletados nas caldeiras. No entanto, os medidores físicos estão sujeitos a falhas em seu funcionamento, atrasos e medições descalibradas, que podem comprometer toda a produção dependente desse recurso.

Surge como opção a utilização de um sensor virtual desenvolvido a fim de proporcionar vantagens como a possibilidade de substituição do sensor físico, além da possibilidade do seu uso concomitante com o mesmo, a fim de se comparar o valor real e previsto e detectar rapidamente um possível mau funcionamento. Em casos como este, onde existe toda uma rede de dependência do vapor gerado, o mau funcionamento fornecendo medidas irreais pode ser ainda mais perigoso do que a parada completa do sensor, pois pode levar a tomadas de decisão erradas e pode ser de mais difícil detecção.

1.2 Objetivos

Este trabalho tem como objetivo geral o desenvolvimento de um sensor virtual para a previsão da quantidade de vapor gerada a partir de dados de sensores secundários presentes na caldeira. Para isto, será utilizado uma Rede Neural Artificial do tipo *Multilayer Perceptron* no intuito de desenvolver um modelo regressivo que consiga estimar esta variável tão importante para a produção da planta analisada. Como objetivos específicos, o trabalho se propõe a implementar várias configurações diferentes para a Rede Neural a fim de permitir a comparação de seu desempenho, bem como a implantação do sensor virtual através de uma aplicação *web* de visualização gráfica em tempo real que permite a visualização dos pontos medidos minuto a minuto. Com a aplicação, será possível tanto a previsão de um único valor estimado de vapor gerado dadas as quantidades de combustível, como simular seu funcionamento no ambiente de produção através de uma ferramenta de monitoramento gráfica.

1.3 Estrutura do trabalho

O presente trabalho se encontra dividido em uma estrutura capitular. O próximo capítulo visa apresentar as bases de todos os temas conceituais que foram utilizados no

desenvolvimento deste projeto. De uma forma geral são abordados assuntos referentes à reação de combustão, caldeiras industriais, sensores virtuais, Redes Neurais Artificiais, além de ferramentas como a linguagem *Python* e seus *frameworks*, e o banco de dados de séries temporais onde os dados de medidores da siderúrgica são armazenados. O capítulo seguinte mostra a parte prática do projeto e explicita a metodologia que foi empregada na obtenção dos resultados. Por último, os dois capítulos finais são responsáveis por apresentar os resultados obtidos e concluir o trabalho.

2 Referencial teórico

Este capítulo tem por objetivo apresentar a base conceitual de todos os elementos que foram utilizados no desenvolvimento deste trabalho a fim de fornecer uma fundamentação teórica daquilo que foi realizado.

2.1 Combustão

Primeiramente é necessário apresentar os princípios regentes desse processo tão importante no meio industrial e que, de certa forma, é o cerne deste trabalho juntamente com as Redes Neurais Artificiais e os Sensores Virtuais.

Como apresentado por Turns (2013), o processo de combustão está disseminado na vida diária das pessoas e tem fundamental importância em áreas como a geração de calor, transporte, geração de energia, entre outros. De acordo com Mcallister, Chen e Fernandez-Pello (2011), a combustão é essencialmente importante no setor industrial na produção de ferro gusa e aço por exemplo, além de estar presente nos fornos de aquecimento, tratamentos térmicos, aquecedores de fluidos, secadores, estufas e caldeiras industriais, que são o foco deste trabalho.

Para o desenvolvimento da proposta, é importante a explicação do processo de forma geral antes de adentrar no campo específico das caldeiras industriais. Para tal, será abordada a relação estequiométrica presente no processo, bem como a importância e efeito dos combustíveis e a sobra de gás oxigênio nos produtos da reação.

2.1.1 Processo

O centro do processo de combustão se encontra em dois principais componentes, chamados de combustível e comburente. Essa distinção pode ser explicada pelo fato de que o combustível é a substância que libera calor quando reage com um comburente (TURNS, 2013). De acordo com Vandagriff (2001), na maioria dos casos o próprio ar terrestre é utilizado como comburente do processo, sendo formado majoritariamente por oxigênio e nitrogênio, bem como por uma porcentagem baixa de outros gases. A escolha do combustível, no entanto, depende muito do processo com o qual está associado à combustão.

2.1.2 Combustíveis

Como é mostrado por Mcallister, Chen e Fernandez-Pello (2011), a escolha do combustível a ser utilizado depende de vários fatores ligados à aplicação deste processo.

Entre eles, pode-se destacar o conteúdo energético disponível por volume de combustível, segurança, combustibilidade e custo do combustível disponível. Levando isto em conta, os combustíveis podem ser classificados de diferentes maneiras, mas para este trabalho eles foram classificados de acordo com a natureza de seu estado físico. De acordo com Ragland, Bryden e Kong (2011), os combustíveis podem ser:

a) Combustíveis Gasosos: São os combustíveis que são transportados na forma de gás. Entre eles, podem ser destacados o gás natural e o GLP (Gás Liquefeito de Petróleo). O gás natural se encontra disponível na natureza comprimido em rochas porosas e em formações de xisto abaixo do solo, enquanto o GLP, apesar de poder ser originado de diversos processos, pode ser entendido como uma mistura de etano, propano e butano oriundos de plantas de processamento de gás natural (MCCALLISTER; CHEN; FERNANDEZ-PELLO, 2011). Algumas propriedades importantes para os combustíveis gasosos são mostradas por Ragland, Bryden e Kong (2011) e incluem densidade, valor calorífico, análise volumétrica, entre outros;
b) Combustíveis líquidos: São combustíveis transportados em forma líquida, mas que são vaporizados para poderem ser utilizados no processo de combustão. A maioria dos combustíveis líquidos se origina a partir do óleo cru, substância encontrada na natureza formada por hidrocarbonetos, minerais e que podem possuir porcentagens de oxigênio, enxofre e nitrogênio (RAGLAND; BRYDEN; KONG, 2011). Como mostrado por McCallister, Chen e Fernandez-Pello (2011), a gasolina e o diesel são combustíveis líquidos que surgem a partir do processo de refinamento do óleo cru. As principais propriedades analisadas neste tipo de combustível são poder calorífico, viscosidade, densidade relativa, ponto de vaporização;
c) Combustíveis sólidos: Entre os combustíveis sólidos encontram-se a madeira, o carvão e a turfa, por exemplo. Suas principais propriedades para análise são o poder calorífico, capacidade de moagem, temperatura de incineração.

Assim, observa-se que a escolha dos combustíveis pode variar bastante, enquanto na grande maioria dos casos o gás oxigênio presente no ar é utilizado como a substância comburente do processo.

2.1.3 Estequiometria da reação

Como foi discutido, o processo de combustão consiste basicamente na reação química ocorrida na interação entre elementos combustíveis e comburentes, com o intuito de gerar

calor para um determinado processo. Também foi discutido que o principal elemento comburente presente nesse processo é o ar terrestre, formado principalmente pelos gases oxigênio e nitrogênio. Além disso, cada processo pode ter seu principal tipo de combustível, dependendo da aplicação envolvida. Dito isso, e seguindo a relação estequiométrica mostrada por Ragland, Bryden e Kong (2011) na Equação 2.1, onde é balanceado um processo de combustão do gás metano, é possível visualizar as relações que surgem entre as quantidades de combustível, comburente e produto da reação. O metano é representado pelo CH_4 enquanto o oxigênio, nitrogênio e água são respectivamente O_2 , N_2 e H_20.

$$CH_4 + a(O_2 + 3.76N_2) \rightarrow b(CO_2) + c(H_20) + 3.76a(N_2) \quad (2.1)$$

Através de cálculos estequiométricos, é visível pela Equação 2.2 quais seriam os coeficientes utilizados no processo de combustão do CH_4.

$$CH_4 + 2(O_2 + 3.76N_2) \rightarrow CO_2 + 2H_20 + 7.52N_2 \quad (2.2)$$

De uma forma ainda mais generalista, McCallister, Chen e Fernandez-Pello (2011) apresentam uma estequiometria genérica para o processo de combustão de um hidrocarboneto qualquer de forma $C_\alpha H_\beta O_\gamma$, tendo o ar terrestre como comburente, conforme a Equação 2.3.

$$C_\alpha H_\beta O_\gamma + (\alpha + \frac{\beta}{4} - \frac{\gamma}{2})(O_2 + 3.76N_2) \rightarrow \alpha(CO_2) + \frac{\beta}{2}(H_20) + 3.76(\alpha + \frac{\beta}{4} - \frac{\gamma}{2})(N_2) \quad (2.3)$$

Um importante conceito derivado da reação de combustão acima é a quantidade de ar necessário para que o processo ocorra, chamada de taxa estequiométrica de ar (MCCALLISTER; CHEN; FERNANDEZ-PELLO, 2011).

2.1.4 Relação entre quantidade de Ar e Combustível

A eficácia de uma reação de combustão está diretamente ligada à quantidade de ar com a qual irá reagir o combustível. De acordo com Vandagriff (2001), a combustão perfeita ou estequiométrica é aquela onde a totalidade do oxigênio contido no ar comburente é oxidada na reação. Qualquer quantidade que seja fornecida acima desta quantia ótima é chamada de excesso de ar. É importante ressaltar que deve haver um balanceamento no fornecimento desse excesso de ar. Como o cálculo da quantidade estequiométrica leva em

consideração algumas condições que muitas vezes não se repetem no mundo real, o excesso de ar é importante como uma margem de segurança para que haja uma combustão completa no processo. No entanto, esta quantidade deve ser apenas suficiente para garantir a eficiência da combustão e não mais do que isto.

Seguindo a análise apresentada por McCallister, Chen e Fernandez-Pello (2011), pode-se definir alguns parâmetros importantes que ajudam na definição da quantidade de excesso de ar, conforme a Equação 2.4. Em primeiro lugar, o parâmetro responsável por relacionar a quantidade de ar e de combustível presente na mistura na entrada da reação pode ser definido através de uma razão de massas.

$$f = \frac{m_f}{m_a} \tag{2.4}$$

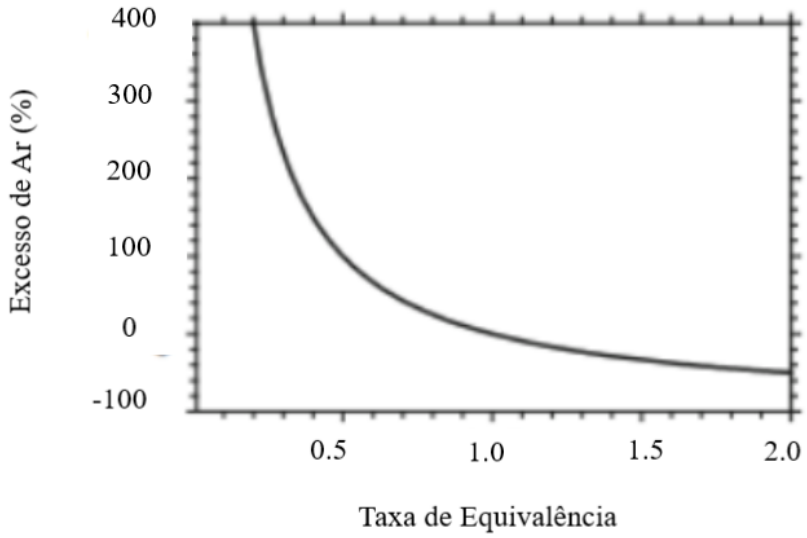
A variável *m* representa a massa dos componentes, estando a massa do combustível no numerador e a massa de ar no denominador. É possível modificar essa fórmula utilizando também as variáveis estequiométricas e chegar a um parâmetro auxiliar f_s, que se utilizado como divisor de f, dá origem ao termo ϕ , conhecido como taxa de equivalência. A partir de seu valor, a mistura pode ser classificada como mistura rica se $\phi > 1$, mistura pobre se $\phi < 1$ ou mistura estequiométrica se $\phi = 1$.

Esse parâmetro é importante pois fornece um dos modos de calcular a porcentagem de excesso de ar utilizada. Como foi dito, essa porcentagem fornece um valor que representa a quantidade de ar em excesso em relação à quantidade estequiométrica que está sendo utilizada no processo.

Seu valor pode ser obtido da seguinte forma mostrada pela Equação 2.5 e ilustrada pela Figura 2.1.

$$\%EA = 100 \left(\frac{1-\phi}{\phi}\right) \tag{2.5}$$

Figura 2.1 - Excesso de ar.



Fonte: Adaptado de Mcallister, Chen e Fernandez-Pello (2011)

A importância do entendimento do excesso de ar na reação de combustão é mostrada por Tillman (1981 citado por MENGHINI et al., 2008) que diz que o seu valor influencia diretamente na eficiência da combustão e na temperatura da chama, que por sua vez, influenciam em fatores como quantidade e qualidade de calor liberados pelo processo e consequentemente no vapor a ser gerado.

De acordo com Vandagriff (2001), o excesso de ar resulta em uma chama mais curta e fraca, e que esse oxigênio em excesso não é consumido no processo e por isso acaba por absorver e dissipar parte da quantidade de calor que poderia ser útil, diminuindo a eficiência do processo. De uma forma contrária, se pouco oxigênio é fornecido, a chama obtida é alongada e produz muita fumaça. Como não há quantidade suficiente de oxigênio para queimar todo o combustível, há formação de sobras de monóxido de carbono, fumaça e fuligem.

2.2 Caldeiras Industriais

As caldeiras são muito importantes nas atividades de uma indústria porque têm como principal objetivo a geração de energia para ser utilizada em outros processos. De acordo com Elkelawy e Alm Edin (2022), a caldeira é um tipo de recipiente fechado onde uma quantidade de água é aquecida através de um processo de combustão de modo a gerar vapor que é utilizável para transferir calor para um outro processo.

De uma forma geral, o sistema de caldeiras pode ser dividido em três subsistemas: sistema de alimentação de água, sistema de vapor e sistema de combustível. O primeiro deles é responsável pelo fornecimento de água para a caldeira. O segundo tem o encargo de coletar o vapor produzido e controlar o seu fornecimento. Existem várias válvulas responsáveis por regular os controles exercidos por cada subsistema. Por último, o sistema de combustível contém todo o equipamento necessário para o processo de combustão e fornecimento de calor (ELKELAWY; ALM EDIN, 2022).

2.2.1 Tipos de Caldeiras

A partir do que é apresentado por Botelho e Bifano (2015), é possível fazer a classificação das caldeiras em diversas categorias. A primeira categoria diz respeito à sua disposição interna. Em relação a isso, podem ser de dois tipos:

a) Fogotubular: Como o próprio nome sugere, neste tipo de caldeira são os gases quentes que estão contidos em tubos, sendo que estes estão circundados por água;
b) Aquatubular: De maneira contrária à anterior, neste caso a água circula dentro de serpentinas e é aquecida pelos gases quentes ao seu redor, responsáveis por fazer a geração de vapor.

Como mostrado por Elkelawy e Alm Edin (2022), existem algumas vantagens no sistema de caldeiras onde a água circular é envolvida pelos gases quentes. Entre elas, podem ser citadas algumas tais como:

a) Geração de vapor mais rápida, com capacidade de evaporação e pressão de vapor maiores;
b) Superfícies de aquecimento mais efetivas devido ao contato dos gases quentes com o fluxo de água;
c) Maior eficiência da combustão, já que o espaço maior facilita a combustão completa do combustível;
d) Os danos provocados por vazamento de água são bem menos críticos.

No entanto, também são apresentados alguns deméritos deste tipo de caldeira em relação ao tipo fogotubular:

a) Custos de manutenção são mais altos;
b) Caso haja um problema na alimentação de água, ainda que por um curto período de tempo, é provável que haja superaquecimento da caldeira.

Outra forma de fazer esta classificação é em relação ao tipo de combustível utilizado. São esses combustíveis que, queimados no processo de combustão em contato com o ar, irão gerar o calor necessário para os outros processos de trabalho. De acordo com Botelho e Bifano (2015), os principais tipos utilizados em caldeiras industriais são madeira, cavaco, bagaço de cana, GLP, gás natural, óleo combustível e diesel.

2.2.2 Combustão em Caldeiras Industriais

Como a função principal das caldeiras está diretamente relacionada ao fornecimento de energia térmica a ser usada em outros processos, o processo de combustão é de vital importância para seu funcionamento. É através dele que ocorre a queima do combustível fornecido e a consequente liberação de calor. A caldeira em questão que está sendo analisada, utiliza uma mistura de combustíveis formada principalmente por Gás de Aciaria, Gás de Alto Forno, Gás de Coqueria e Gás Natural além do Nitrogênio. Para este último já foi feita uma explicação sobre sua origem, mas para o restante é interessante entender a sua formação. Sua classificação se difere logo de cara do Gás Natural devido a sua origem. Enquanto o Gás Natural, como o próprio nome já diz, tem origem na natureza, os outros são subprodutos originados de processos na siderurgia:

a) Gás de Aciaria: É também chamado de LDG (*Linz-Donawitz gas*). Este gás é gerado durante o sopro de oxigênio no convertedor e é captado durante este processo para ser utilizado em outro (CSP, 2021). É formado majoritariamente por monóxido de carbono, mas possui também gás carbônico, nitrogênio e pequenas quantidades de hidrogênio e oxigênio (COSTA, 2019);
b) Gás de Alto Forno: Também conhecido como BFG (*Blast Furnace Gas*). Ele é originado a partir da reação existente no processo de obtenção do ferro-gusa, que acontece entre o minério de ferro e o coque (FERREIRA, 2015). Sua composição é formada principalmente por nitrogênio, dióxido de carbono, monóxido de carbono, além de possuir hidrogênio e gás sulfídrico em menores quantidades;
c) Gás de Coqueria: Pode ser encontrado sob a sigla COG (*coke oven gas*) e é oriundo do resultado da transformação do carvão em coque em um processo siderúrgico. Sua composição se dá em grande parte de hidrogênio, gás metano e monóxido de carbono.

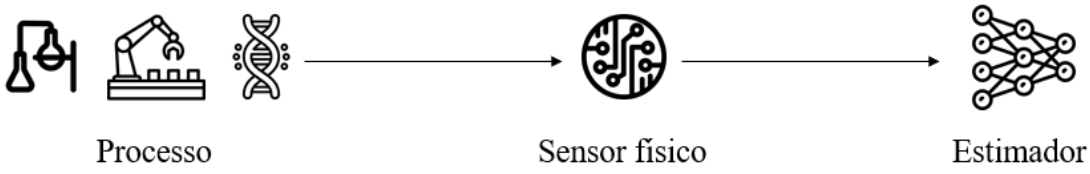
Desse modo, observa-se a importância do processo de combustão nas caldeiras industriais e como existem várias variáveis envolvidas neste processo, influenciando diretamente na eficiência da reação e na formação dos produtos finais.

2.3 Sensores Virtuais

De uma forma geral, pode ser entendido como sensor virtual (*soft sensor)* um *software* com um modelo matemático capaz de estimar uma variável de interesse para a medição, a partir de outras variáveis secundárias.

Sua utilização está vinculada a processos em que o sensor físico, por algum motivo, não pode ser instalado. Como mostrado por Lotufo e Garcia (2008), a impraticabilidade de um sensor físico pode ser devida ao alto custo para tal, à inexistência do instrumento de medição para a variável de interesse, aos erros de medição inerentes em um instrumento físico que podem ser intoleráveis dependendo da aplicação, atrasos na obtenção de medidas críticas, entre outros. Unido a isto, Fortuna et al. (2007) mostra também que os sensores virtuais são muito vantajosos pois permitem o trabalho em paralelo com um sensor físico aumentando a confiabilidade da coleta de dados, permitem uma fácil implementação em *hardwares* como microcontroladores e permitem a estimativa de dados em tempo real. O esquemático da Figura 2.2 ilustra a integração do sensor virtual com o processo analisado.

Figura 2.2 - Esquemático *Soft Sensor.*

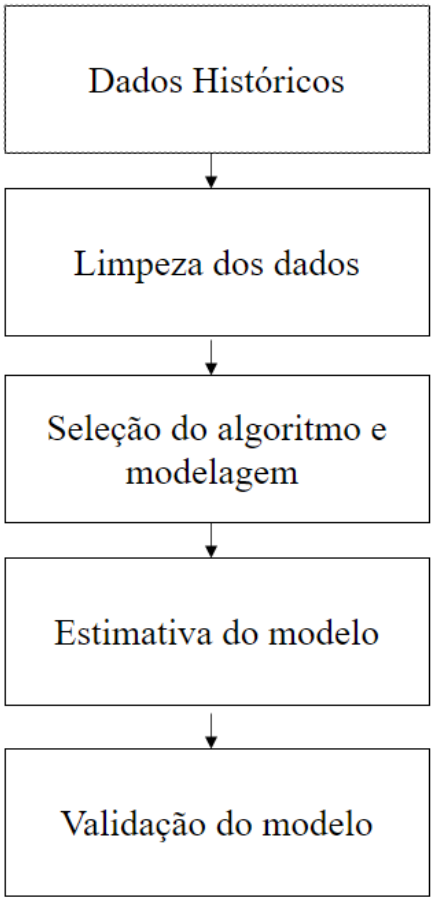


Fonte: Adaptado de Luttman et al (2012)

2.3.1 Métodos Utilizados

De acordo com Fortuna et al. (2007), existe um esquema organizado em etapas que elucida como pode ser feito o desenvolvimento de um processo de identificação de sistemas, e que pode ser aplicado para um sensor virtual, como mostrado na Figura 2.3. Obviamente, depende muito do motivo pelo qual está ocorrendo a virtualização do sensor, mas a estrutura básica de desenvolvimento, pode ser entendida como se segue:

Figura 2.3 - Sequência de desenvolvimento do *Soft Sensor.*



Fonte: Adaptado de Fortuna et al. (2007)

A princípio, os modelos de sensores virtuais baseados em dados necessitam de uma grande base de dados históricos para análise e processamento. Esses dados podem ser, por exemplo, das variáveis secundárias que posteriormente serão utilizadas para a estimativa da variável de interesse monitorada no processo. As partes mais importantes, no entanto, se concentram nos três últimos blocos do esquemático apresentado, onde a modelagem visa tornar possível o conhecimento da natureza do processo em análise.

De acordo com Lotufo e Garcia (2008), quando o comportamento do sistema é descrito através de uma formulação matemática que advém de um completo conhecimento da sua natureza, tem-se a chamada modelagem “caixa-branca”. Porém, como mostra Fortuna et al. (2007), a recorrente complexidade presente em processos industriais unida à disponibilidade de dados históricos, força com que essa modelagem seja feita através de outra abordagem, a da chamada “caixa-preta”. Através dessa abordagem, é possível que seja feita a

estimativa de um modelo que vise criar uma função matemática genérica de relacionamento entre as variáveis envolvidas no processo.

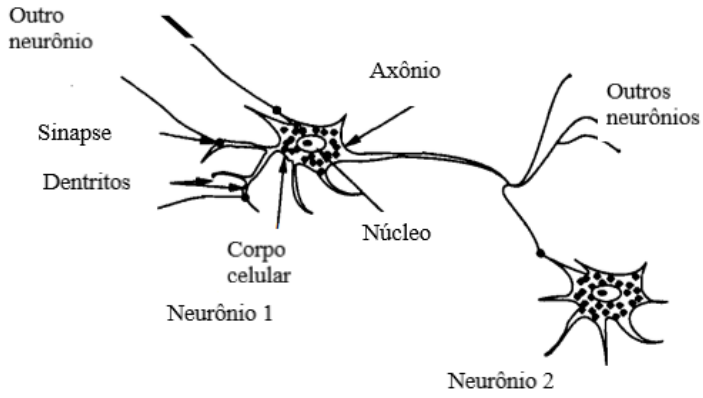
Em relação aos possíveis modelos utilizados para cumprir este papel, Fortuna et al. (2007) e Gonzalez (1999) mostram que existe uma grande variedade para isto. Desde modelos baseados em regressão, como ARMA e NARMAX, passando pelo Filtro de Kalman e PCA (*Principal Component Analysis*) e chegando até as Redes Neurais muito utilizadas em trabalhos atuais e que foi utilizada neste trabalho.

2.4 Redes Neurais Artificiais

Antes de entrar na explicação do algoritmo propriamente dito, é importante conhecer a motivação por trás de sua existência. Assim como muitos conceitos presentes na Ciência de um modo geral, seu cerne se encontra na observação da natureza biológica. De acordo com Haykin (2007) a principal motivação por trás do surgimento das Redes Neurais Artificiais se baseia na admiração do poder de processamento do cérebro humano, e no modo como isso se diferencia de um processamento computacional. Desse modo, têm-se que os primeiros passos dados para a formação do que hoje se conhece como Rede Neural Artificial vieram da tentativa da imitação da estrutura de uma Rede Neural Natural.

A unidade básica de uma Rede Neural é uma estrutura singular chamada de neurônio, ilustrada na Figura 2.4. Como mostra Yegnanarayana (2009), o corpo celular de um neurônio se conecta à extremidade do outro através dos dendritos. Desse modo, as informações são transmitidas através dessa junção caso um potencial elétrico formado ultrapasse um determinado valor limiar.

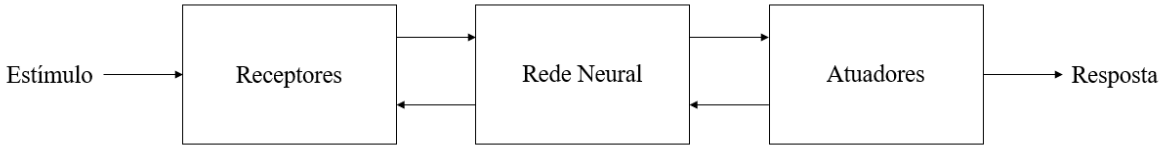
Figura 2.4 - Neurônio Biológico.



Fonte: Adaptado de Yegnarayana (2009)

De acordo com Haykin (2007), o funcionamento do cérebro humano pode ser simplificado de uma forma geral, como mostrado na Figura 2.5. Sendo os neurônios a unidade básica, que unidas formam uma Rede Neural, pode-se pensar em um diagrama de blocos de processamento, onde os estímulos são processados pela unidade central, nesse caso o cérebro, e geram respostas.

Figura 2.5 - Fluxo de Informações.



Fonte: Haykin (2007).

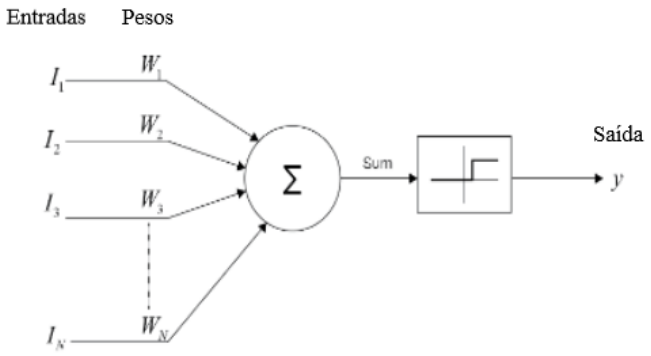
As Redes Neurais Artificiais se comportam de um modo parecido, recebendo estímulos de suas unidades básicas, também chamadas de neurônios, e processando essa informação de modo a propagá-la para os próximos níveis.

2.4.1 Neurônio matemático

Para a formulação de uma Rede Neural completa, é necessário que antes seja feito o entendimento de sua unidade básica de funcionamento. Uma Rede Neural Artificial pode ser entendida como um modelo extremamente simplificado de uma Rede Neural Biológica. Sendo assim, sua unidade fundamental também pode ser chamada de neurônio, unidades processadoras que unidas entre si formam uma Rede Neural (YEGNANARAYANA, 2009)

De acordo com Arbib (1995), é preciso entender os conceitos da espécie mais simples de neurônio matemático proposto por McCulloch e Pitts (1943). Este modelo primordial propunha um limiar binário para a transmissão de informação, que viria a partir de uma soma ponderada de outros sinais. As semelhanças entre as estruturas básicas do neurônio biológico com o matemático começam a ficar mais evidentes a partir da Figura 2.6.

Figura 2.6 - Neurônio matemático.



Fonte: Rothman (2018)

Pode-se observar a associação entre as entradas do neurônio matemático com os impulsos elétricos que alimentam um neurônio biológico. Outro parâmetro importante nesse modelo primordial se encontra no somatório das entradas ponderadas, a junção aditiva que produz o valor de ativação (YEGNANARAYANA, 2009). Esse valor é importante pois representa a informação conjunta das entradas, e é repassado para os próximos neurônios caso atinja um determinado limiar.

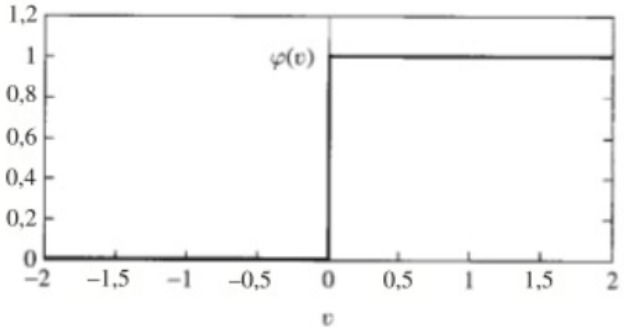
Esta generalização pode ser entendida a partir da Equação 2.6 , adaptada de Jain, Mao e Mohiuddin (1996).

$$y = \theta \left(\sum_{j=1}^{n} w_j x_j \right) \tag{2.6}$$

A variável θ é um exemplo de função degrau, que pode ser entendida da seguinte forma, conforme a Equação 2.7 e Figura 2.7.

$$f(x) = 1 \; se \; x \geq 0, \; 0 \; se \; x < 0 \tag{2.7}$$

Figura 2.7 - Função Degrau.

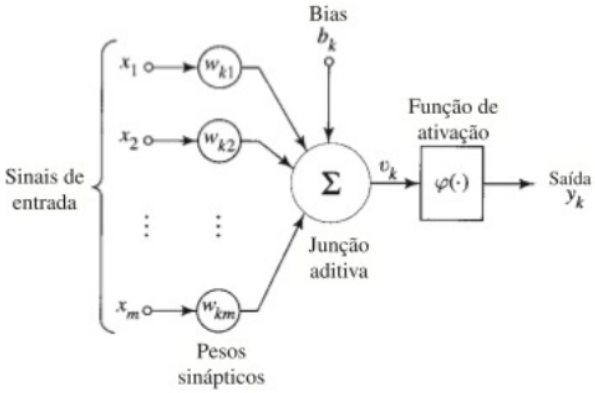


Fonte: Haykin (2007)

Além disso, w_j é o peso atrelado à entrada x_j. Fica evidente desse modo que as entradas podem ter uma influência maior ou menor a depender da sua ponderação. Inclusive, podem ser consideradas como excitatórias, se tiverem valores positivos, ou inibitórias se negativas (JAIN; MAO; MOHIUDDIN, 1996) .

Além disso, em alguns casos pode existir o parâmetro conhecido como *bias,* mostrado na Figura 2.8. Este pode ser entendido como uma influência positiva ou negativa adicionada à junção de ativação que contribui ou não para que a soma ponderada atinja o limiar atingido pela função de ativação.

Figura 2.8 - Neurônio matemático detalhado.



Fonte: Haykin (2007)

O neurônio matemático proposto por McCulloch e Pitts (1943) foi importante pois permitiu que outros modelos mais complexos surgissem. Mas é importante ressaltar que fazem muitas simplificações que não estão presentes nos neurônios biológicos e que muitas vezes não são válidas nos casos práticos onde as Redes Neurais Artificiais são aplicadas.

Uma dessas simplificações se encontra na função de ativação, considerada a princípio como um simples degrau, mas que pode assumir outras variadas formas.

2.4.2 Funções de Ativação

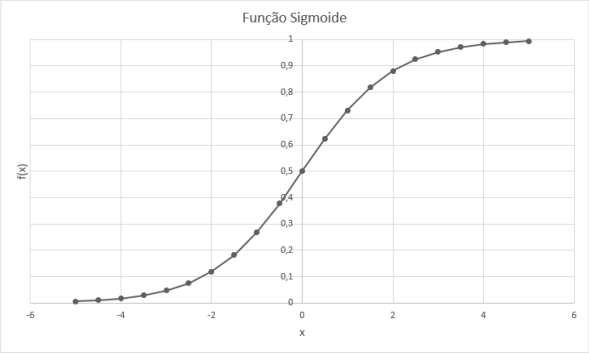
De acordo com Arbib (1995), as funções de ativação têm um papel fundamental nas Redes Neurais Artificiais pois são responsáveis por validar a transmissão de informação de uma camada da Rede para a camada subsequente. No caso dos neurônios simples, como a estrutura de McCulloch e Pitts (1943), são as funções de ativação as responsáveis por definir a resposta obtida pelo processamento do neurônio. Como mostrado por Arbib (1995) existem alguns tipos principais de função de ativação utilizados nas Redes Neurais Artificiais. São elas:

a) Função sigmóide: amplamente utilizada nas Redes Neurais Artificiais. É uma função que cresce com suavidade e possui algumas características assintóticas interessantes, sendo uma função logística com uma faixa de valores assumidos entre 0 e 1 (BISHOP , 2006). Sua função matemática de origem é definida como a mostrada na Equação 2.8;

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.8}$$

E graficamente, a relação entre as variáveis dependente e independente pode ser vista na Figura 2.9.

Figura 2.9 - Função Sigmóide.

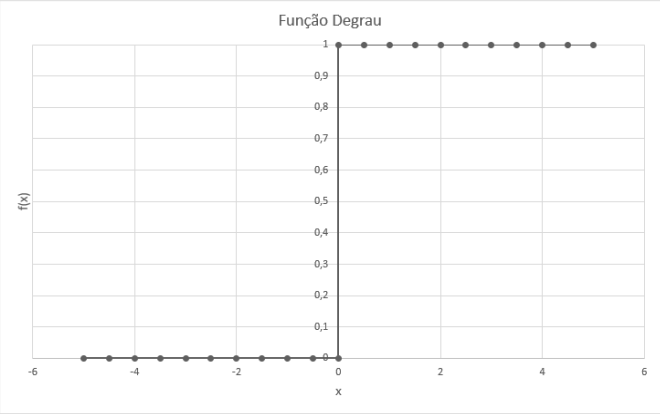


Fonte: Do Autor (2022)

b) Função degrau: uma das mais simples funções empregadas, que aparece inclusive no modelo primordial de neurônio proposto por McCulloch e Pitts (1943). Sua definição em Sharma, Sagar, Sharma, Simone e Athaya (2020) mostra uma função que tem um

valor inicial zerado. É definido um determinado valor limiar a partir do qual ela passa a assumir o valor 1 infinitamente. Em alguns casos mais comuns, esse valor limiar é o próprio 0. Sua equação regente, bem como sua resposta gráfica se encontram na Equação 2.7 e Figura 2.10.

Figura 2.10 - Função Degrau simulada.

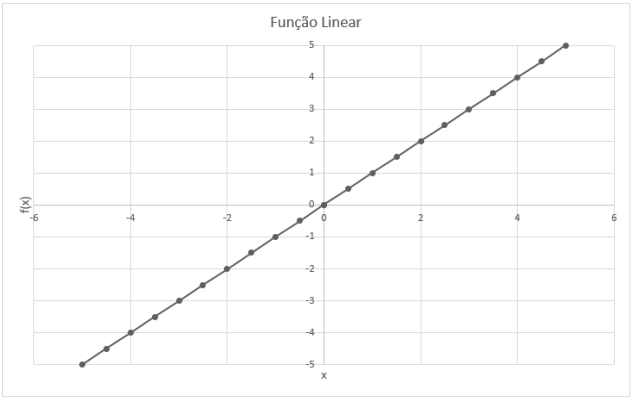


Fonte: Do Autor (2022)

c) Função linear: matematicamente, é uma função cuja saída é diretamente proporcional a sua entrada, como mostrado na Equação 2.9 e Figura 2.11. De uma forma geral:

$$f(x) = ax \text{ , onde } a \text{ é uma constante.} \quad (2.9)$$

Figura 2.11 - Função Linear.



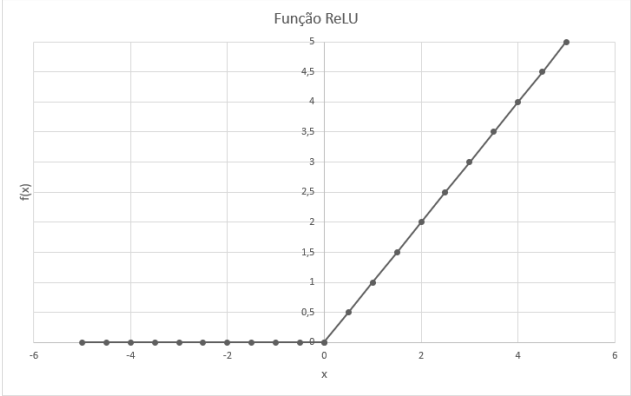
Fonte: Do Autor (2022)

d) Função ReLU: amplamente utilizada em muitas Redes Neurais Artificiais. Seu comportamento é bastante parecido com a função de ativação linear. Sua diferença está no fato de que os neurônios que recebem informação representada por um valor

negativo, não transmitem essa informação para a camada subsequente (SHARMA, Sagar; SHARMA,Simone; ATHAYA, 2020). Matematicamente isso é mostrado na Equação 2.10 e graficamente na Figura 2.12:

$$f(x) = max(0, x) \tag{2.10}$$

Figura 2.12 - Função ReLU.



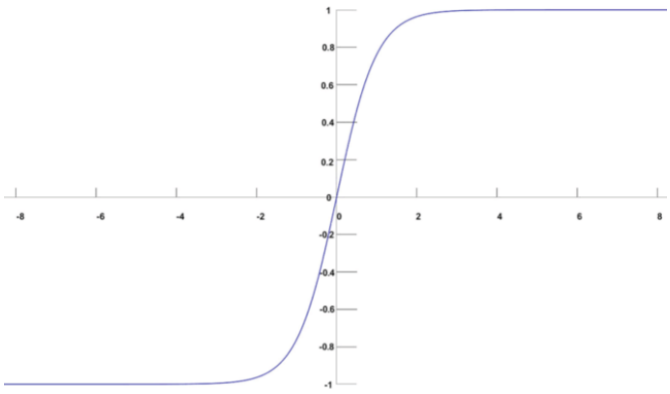
Fonte: Do Autor (2022)

- Função *softmax:* De acordo com Arbib (1995), é utilizada principalmente em problemas de classificação que fogem da separação binária, onde é utilizada a função sigmoide. Como é dito em Sharma,Sagar, Sharma,Simone e Athaya (2020), a função *softmax* pode ser entendida como uma combinação de várias funções sigmoide. Sua expressão matemática pode ser definida conforme a Equação 2.11:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}, \tag{2.11}$$

onde a saída da função pode ser definida como uma soma de probabilidades de cada entrada pertencer a uma determinada classe. O termo e^{z_j} representa a exponencial a qual todos os valores do vetor de entrada são submetidos. O somatório no denominador é responsável pela normalização e garante que a soma dos valores seja igual a 1. Além disso, o K representa a quantidade de classes do problema (WOOD, 2022). Graficamente isto pode ser visto na Figura 2.13.

Figura 2.13 - Função *softmax.*



Fonte: Do Autor (2022)

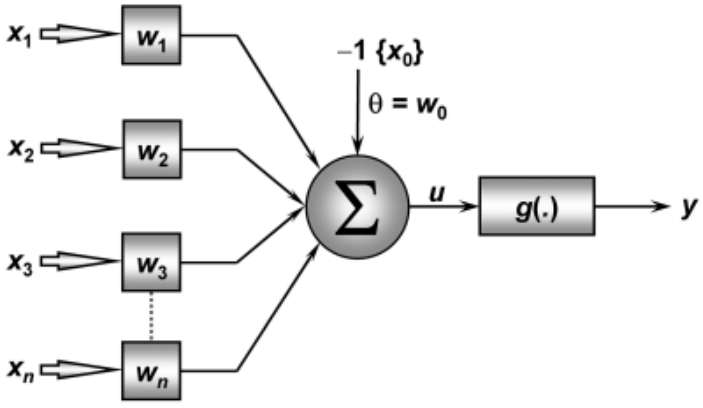
2.4.3 *Perceptron*

Segundo Haykin (2007), pode-se entender o *perceptron* como a unidade mais simples daquilo que um dia se tornaria uma Rede Neural Artificial, geralmente composta por múltiplos *perceptrons,* sendo utilizado para a resolução de problemas de classificação desde que sejam linearmente separáveis. A estrutura formativa dessa Rede Neural segue de perto o que foi apresentado como sendo o neurônio matemático e seus parâmetros associados. Isso pois, sua camada de processadores consiste justamente de um único neurônio.

O *perceptron* foi concebido inicialmente por Rosenblatt (1958), tendo certos elementos interligados para o seu funcionamento. De acordo com Silva, Spatti e Flauzino (2010), os sinais de entrada para a Rede Neural em questão provinham de fotocélulas, sendo então ponderados por elementos resistivos e somados posteriormente. Sua principal inspiração se encontrava na retina.

De uma forma geral, o *perceptron* possui então pesos ajustáveis associados aos valores de suas entradas e um peso responsável por ponderar uma entrada dita unitária, conhecido como *bias,* conforme a Figura 2.14.

Figura 2.14 - *Perceptron.*



Fonte: Silva, Spatti e Flauzino (2010)

Como demonstrado por Silva, Spatti e Flauzino (2010), é possível fazer uma análise matemática simples para evidenciar a principal característica do *perceptron,* seu poder de separar linearmente. Considerando apenas duas entradas ponderadas por seus pesos e levando em conta a existência de um valor de *bias,* pode-se inferir que:

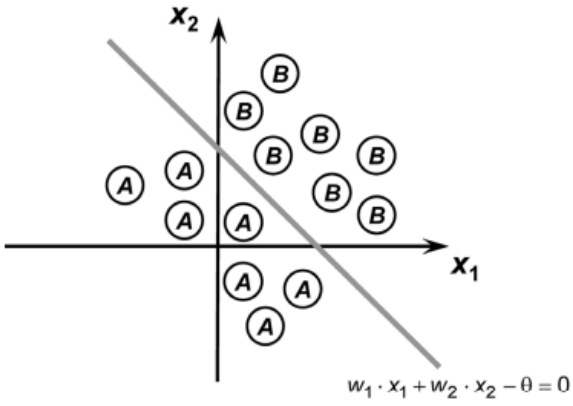
a) O valor de saída de uma função de ativação do tipo degrau será 1 se $w_1 x_1 + w_2 x_2 - \theta \geq 0$;
b) O valor de saída de uma função de ativação do tipo degrau será 0 se $w_1 x_1 + w_2 x_2 - \theta < 0$.

Chega-se então em uma representação de primeiro grau conforme a Equação 2.12.

$$w_1 x_1 + w_2 x_2 - \theta = 0 \tag{2.12}$$

Isso mostra o caráter classificatório binário do *perceptron,* cuja equação obtida é capaz de separar as amostras em duas classes diferentes. Isto fica mais evidente a partir da observação da Figura 2.15 que mostra a classificação de amostras.

Figura 2.15 - Equação classificadora.



Fonte: Silva, Spatti e Flauzino (2010)

2.4.4 Aprendizagem

Como é dito por Jain, Mao e Mohiuddin (1996), a capacidade de aprender é uma característica fundamental da inteligência. Essa propriedade define a Rede Neural Artificial, dando a ela a capacidade de ter o seu desempenho aperfeiçoado através da aprendizagem (HAYKIN, 2007).

Como já foi mostrado, as Redes Neurais Artificiais são formadas por um conjunto de parâmetros que definem o seu funcionamento. Entre esses parâmetros estão os seus pesos e o *bias,* que também pode ser considerado um tipo específico de peso. A partir da definição desses parâmetros, Haykin (2007) fornece uma explicação de como ocorre o processo de aprendizagem de uma Rede Neural Artificial. De uma forma geral, o seu aprendizado ocorre a partir de uma série de ajustes nos seus pesos e *bias*, visando fornecer um mapeamento correto entre suas entradas e saídas.

O *perceptron,* por exemplo, como mostrado por Arbib (1995), é uma Rede Neural que usa um mecanismo de correção de erro para fazer esse ajuste de pesos. Dessa forma, esse conceito une-se a um outro, o da aprendizagem supervisionada. Neste tipo de aprendizagem, a Rede Neural tem acesso à saída desejada dado um conjunto de parâmetros de entrada, e ocorre a comparação entre o valor real desejado e o valor obtido como saída da Rede. De uma forma simplificada a princípio, é possível definir alguns passos de aprendizado seguidos pelo *perceptron* (JAIN; MAO; MOHIUDDIN, 1996):

a) Inicialização aleatória para os valores de pesos e *bias;*
b) Alimentação da rede com vetores de entrada e sua respectiva saída;

c) Atualização dos pesos de acordo com a Equação (2.13).

$$w_j(t+1) = w_j(t) + \eta(d - y)X_J \tag{2.13}$$

Nesta simplificação, o valor para a atualização do peso de índice *j*, isto é, de $w_j(t+1)$, depende do seu valor no momento atual (representado por $w_j(t)$), da diferença entre o valor desejado *d* e o valor obtido *y,* multiplicado pelo valor de sua entrada correspondente representado pelo parâmetro X_J e por um parâmetro conhecido como taxa de aprendizagem, representado por η. Este parâmetro pode assumir valores entre 0 e 1 e mostra como será o processo de convergência dessa Rede Neural (SILVA; SPATTI; FLAUZINO, 2010).

Definidos os parâmetros e o funcionamento simplificado do ajuste de pesos, é importante entender como é feito esse ajuste e qual seu critério de parada.

2.4.5 Descida do gradiente

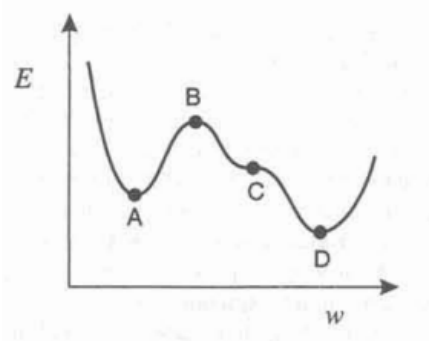
As Redes Neurais Artificiais, enquanto algoritmo de Aprendizado de Máquinas, podem ser entendidas como um algoritmo de otimização. Sabendo que existe uma relação entre os valores dos seus pesos e a saídas obtidas pelo modelo, essa relação pode ser estendida e relacionar os valores escolhidos para os pesos com o erro obtido pelo algoritmo. Dessa forma, é visado que sua estrutura de parâmetros seja otimizada a tal ponto que minimize o valor do erro obtido.

De acordo com Bishop (2006), a atualização de pesos deve ser feita na direção da maior taxa de variação da diminuição do erro, ou seja, na direção do gradiente negativo. Dado que o vetor de pesos em determinado tempo pode ser representado como $w^{(t)}$, então de acordo com a Equação 2.14, onde *E* representa o erro e η é o parâmetro conhecido como taxa de aprendizagem:

$$\Delta w^{(t)} = -\eta \nabla E|_{w^{(t)}} \tag{2.14}$$

É importante a observação da Figura 2.16 e suas implicações para o funcionamento do algoritmo (BISHOP, 2006):

Figura 2.16 - Gráfico erro (E) *versus* peso (*w).*



Fonte: Bishop (2006)

Observa-se que é possível uma relação entre os valores de peso e o valor do erro obtido. Obviamente, a figura mostra um caso hipotético muito específico onde só existe um parâmetro de peso a ser ajustado. A curva obtida mostra vários pontos onde o gradiente da função erro assume o valor 0, satisfazendo a condição mostrada pela Equação 2.15.

$$\nabla E = 0 \tag{2.15}$$

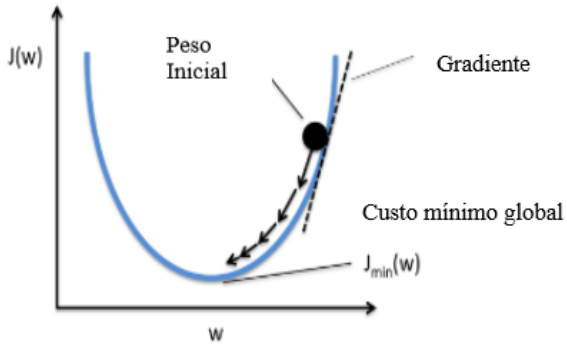
Porém, é interessante fazer a distinção entre os pontos representados pela letra D e os demais. Enquanto o primeiro representa o mínimo global, os outros, com exceção do ponto B, representam os mínimos locais. Visivelmente, é mais vantajoso ao algoritmo que ajuste os pesos de modo a atingir o mínimo global.

Outra análise de interesse diz respeito à escolha do valor de η. De acordo com Haykin (2007), existe uma trajetória percorrida pelos valores de pesos iniciais até atingirem o ponto ótimo. Essa trajetória se divide em algumas categorias quanto à sua convergência:

a) Se a escolha da taxa de aprendizagem é muito pequena, então o caminho percorrido é mais suave;
b) Se a escolha da taxa de aprendizagem é muito grande, então a trajetória é mais abrupta e pode seguir um padrão oscilatório até atingir o mínimo desejado;
c) Se a escolha ultrapassar um valor crítico, então atinge-se a instabilidade.

A Figura 2.17 resume toda a explicação da descida do gradiente em busca do mínimo valor para a função de erro.

Figura 2.17 - Descida do Gradiente.



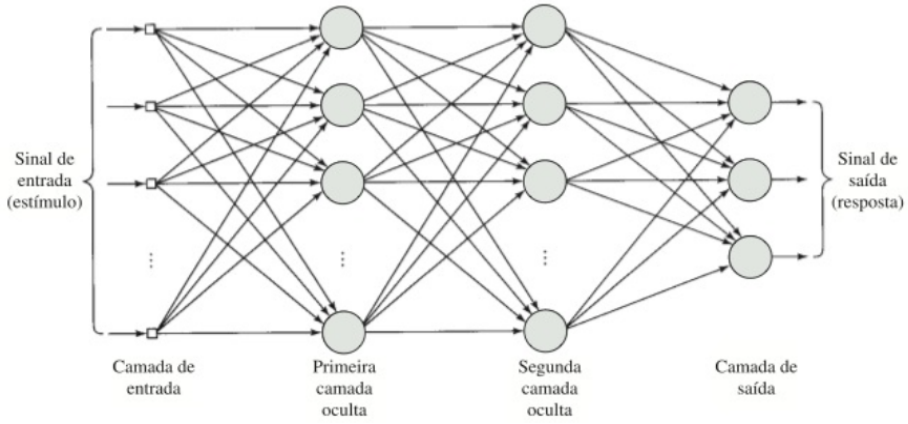
Fonte: Deep Learning Book (2022)

2.4.6 Multilayer Perceptron

Seguindo a estrutura lógica proposta e passando pelas fases iniciais do que hoje é conhecido como Rede Neural Artificial, passando pelo simples neurônio matemático e pelo *perceptron* de camada única, chega-se agora a uma estrutura um tanto mais complexa, o *perceptron* de múltiplas camadas.

Como é dito em Haykin (2007), os *perceptrons* de camadas múltiplas podem ser entendidos como uma estrutura que se divide em algumas camadas. Ao invés do simples neurônio matemático único presente nos *perceptrons* simples, onde a própria saída desse neurônio pode ser entendida como a saída da Rede, neste caso têm-se uma divisão em camadas de entrada, camadas ocultas e camadas de saída. O processamento é feito no sentido de passar as informações para as próximas camadas, como mostrado na Figura 2.18.

Figura 2.18 - *Multilayer Perceptron.*



Fonte: Haykin (2007)

Como é dito por Bishop (2006), o surgimento de Redes Neurais de múltiplas camadas visa permitir um mapeamento mais complexo entre as entradas e saídas, algo que não era possível com o simples *perceptron,* cuja aplicação se restringe a problemas de complexidade inferior.

É importante compreender também de acordo com Bishop (2006) que uma característica fundamental, ainda que de certo modo limitante dos *perceptrons* de múltiplas camadas, é a sua direção da transmissão de informações conhecida como *feed-forwarding* ou alimentação direta. Isso significa que não há laços de realimentação entre os neurônios, característica que pode se fazer presente em outros tipos de Redes Neurais Artificiais.

Como mostrado por Silva, Spatti e Flauzino (2010), esse tipo de Rede Neural é muito utilizado para a resolução de problemas tais como :

a) Aproximação Universal de Funções;

b) Reconhecimento de Padrões;

c) Identificação e Controle de Processos;

d) Previsão de Séries Temporais;

e) Otimização de Sistemas.

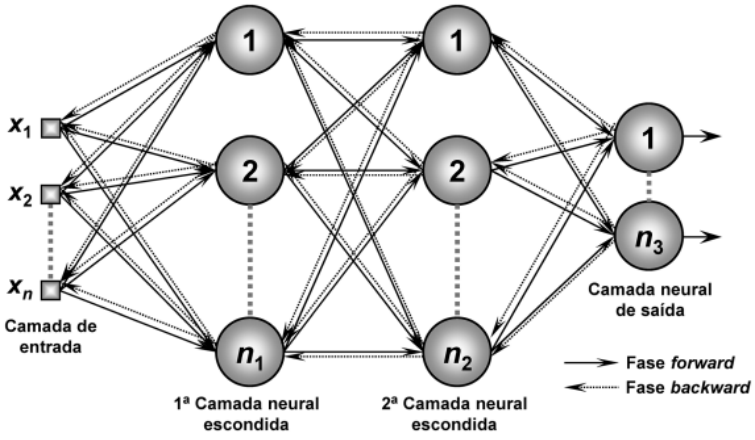
O treinamento desse tipo de Rede Neural se difere um pouco do processo de aprendizagem visto para os *perceptrons* de camadas simples. Para este caso, surge um algoritmo específico para sua estrutura conhecido pela retropropagação do erro através das camadas. Esse algoritmo de treinamento é conhecido como *backpropagation.*

2.4.7 Backpropagation

De uma forma geral, o algoritmo *backpropagation* pode ser entendido por uma divisão entre dois processos principais, como mostrado por Silva, Spatti e Flauzino (2010). A primeira parte consiste na transmissão de informação no sentido direto, também conhecido como fase *forward,* onde são utilizados os valores de pesos atuais para a produção de um valor primário para a saída. Tendo obtidos os valores de saída, e levando em conta que se tem um processo de aprendizagem supervisionada, isto é, onde a Rede Neural tem acesso aos valores reais das saídas, é feito então o cálculo do erro. Esses valores de erro são então propagados de forma reversa para as camadas anteriores da Rede, na fase conhecida como

backward. Justamente neste processo são feitos os ajustes dos pesos entre cada conexão e o consequente treinamento da Rede Neural, como ilustrado na Figura 2.19.

Figura 2.19 - Funcionamento do *backpropagation.*



Fonte: Silva, Spatti e Flauzino (2010)

O funcionamento deste algoritmo pode ser derivado a partir das explicitações em Bishop (2006). Em primeiro lugar, é necessário considerar que neste tipo de Rede Neural cada neurônio irá processar um determinado valor, que chega até ele através de uma soma ponderada de valores de entrada. Esse valor irá sofrer uma propagação ou não, dependendo de como será seu comportamento em relação à função de ativação do neurônio. Isto pode ser representado da seguinte forma de acordo com a Equação 2.16.

$$a_j = \sum_i w_{ji} z_i \tag{2.16}$$

O parâmetro z_i é a entrada de um neurônio, que transmite o valor para o neurônio *j* através dos pesos w_{ji} associados a esta conexão. De modo parecido, a ativação ou entrada do neurônio *j* pode ser entendido como se segue na Equação 2.17.

$$z_j = g(a_j) \tag{2.17}$$

Ainda de acordo com Bishop (2006), o objetivo central é conseguir relacionar a função de erro analisada e os pesos ajustados. Desse modo, é interessante obter uma derivada da função de erro em relação às variáveis de pesos. Como a obtenção do erro depende diretamente da soma ponderada a_j e esta, por sua vez, depende do ajuste dos pesos, através da regra da cadeia pode-se obter a Equação 2.18.

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial a_j}\frac{\partial a_j}{\partial w_{ij}} \tag{2.18}$$

A fim de uma simplificação matemática, pode-se fazer a substituição e chegar na Equação 2.19.

$$\delta_j = \frac{\partial E}{\partial a_j}; z_i = \frac{\partial a_j}{\partial w_{ij}} \Rightarrow \frac{\partial E}{\partial w_{ij}} = \delta_j z_i \tag{2.19}$$

De uma forma simples, a derivada é obtida pela multiplicação de δ associado à saída da conexão pelo valor de *z* associado a entrada da conexão. Para os neurônios de saída, essa validação pode ser obtida conforme a Equação 2.20.

$$\delta_k = \frac{\partial E}{\partial a_k} = g'(a_k)\frac{\partial E}{\partial y_k} \tag{2.20}$$

E de uma forma geral, os valores de δ para os neurônios ocultos podem ser calculados utilizando-se os valores das camadas superiores, chegando à Equação 2.21.

$$\delta_j = g'(a_k) = g'(a_k)\sum_i w_{kj}\delta_k \tag{2.21}$$

Bishop (2006) resume o que foi demonstrado acima com uma série de passos a serem tomados para, de forma recursiva, utilizar o algoritmo *backpropagation:*

a) A primeira parte refere-se à trajetória direta da propagação das informações através das camadas da Rede Neural;
b) Em seguida, obtêm-se os valores de δ_k;
c) Retropropagação dos valores δ encontrados para as camadas anteriores e achá-los para as camadas ocultas.

Os passos acima devem ser repetidos para todos os conjuntos de entradas que alimentam a Rede Neural e os valores obtidos devem ser somados, a fim de que seja encontrada a relação final existente entre o erro obtido e os ajustes de todos os pesos.

2.4.8 Hiperparâmetros

Para a inicialização de uma Rede Neural Artificial aconteça da forma correta, bem como o seu treinamento durante o processamento dos dados, é necessário que alguns parâmetros sejam passados durante a construção do algoritmo. Isso acontece muitas vezes

quando são utilizados *frameworks* para a construção da Rede e a boa escolha destes parâmetros têm grande influência nos resultados obtidos. Em alguns casos isso é feito através de tentativa e erro, onde valores iniciais são supostos para estes hiperparâmetros e, à medida que os resultados vão sendo gerados, eles são ajustados em vista de aumentar os resultados positivos. A escolha ótima é aquela que proporciona resultados melhores. No entanto, também existem algoritmos que fazem esta procura de forma automática e retornam a melhor combinação de valores possível. Para a construção da Rede Neural Artificial, de uma forma geral, os principais parâmetros ajustados são estes:

a) Número de neurônios de entrada: em geral é o primeiro parâmetro definido. Na estrutura da Rede Neural, ele é responsável por alimentar o algoritmo com os dados que estão sendo analisados a partir da base. Ao fornecer os dados para a Rede, o número de neurônios é definido como o número de características que descrevem a base, isto é, os previsores a partir dos quais é gerada a saída;
b) Número de neurônios na camada oculta: o número de neurônios na camada oculta, bem como o número de camadas ocultas pode ser muito variado e, em geral, é feito a partir de tentativa e erro. Em muitos casos é fornecido um número de neurônios intermediário entre o número de entradas e saídas;
c) Função de Ativação: já explicada anteriormente, a função de ativação é o processamento que será aplicado à soma dos dados da camada anterior para propagar a informação para as camadas subsequentes. Na maioria das vezes deve ser especificada para cada camada de neurônios adicionada à Rede;
d) Tamanho do Lote: mais conhecido pelo termo em inglês (*batch size*), este parâmetro está relacionado com a quantidade de registros que serão processados pela Rede Neural antes da sua atualização de pesos. Isto significa que quanto menor o valor, mais vezes os pesos serão atualizados, dado o tamanho da base de dados;
e) Número de Épocas: algumas vezes confundível com o parâmetro anterior, o número de épocas corresponde ao número de vezes em que toda a base de dados será processada pelo algoritmo. Na maioria das vezes, quanto maior o número de épocas, melhor será o treinamento. No entanto esse valor deve ser passado com cautela visto que um número grande de épocas pode ser desnecessário, desperdiçando tempo de processamento;

f) Função de perda: também conhecida como *loss function.* De acordo com a documentação do *Keras*[1]*, framework* de Aprendizagem de Máquinas, a função de perda pode ser entendida como a definição do parâmetro que deverá ser reduzido durante o treinamento da Rede. Alguns exemplos comuns são o Erro Quadrado Médio (*mean squared error*), Erro Absoluto Médio (*mean absolute error*), Entropia Cruzada Binária (*binary cross entropy*) e Entropia Binária Categórica (*categorical cross entropy*). Os dois primeiros são muito comuns em tarefas de regressão, enquanto os dois últimos são muito utilizados para classificação;
g) Métricas: parecido com a função de perda, as métricas são utilizadas para avaliar a performance do modelo. Algumas métricas comumente utilizadas são a Acurácia, Erro Quadrado Médio, Raiz do Erro Quadrado Médio (*root mean squared error*);
h) Otimizadores: os otimizadores estão diretamente ligados a um conceito já apresentado, a taxa de aprendizagem, que define a velocidade de convergência para o ponto de erro mínimo. Em geral, sua utilização está ligada aos *frameworks*, e os mais comumente utilizados são o *Adam, Nadam, RMSProp, Adadelta, Adamax,* entre outros.

2.4.9 Métricas de desempenho

Ao final do processo de construção da Rede Neural, de seu treinamento e consequente previsão dos dados a partir dos pesos obtidos, é necessária a avaliação do modelo. Para isso, existem várias métricas disponíveis e que variam de acordo com o propósito da Rede, isto é, se o seu intuito era a previsão de valores ou classificação dos registros. Para o caso deste trabalho, onde o principal objetivo é a previsão de valores numéricos, surgem algumas métricas proeminentes tais como:

a) Erro Quadrado Médio: muitas vezes aparece com o termo em inglês *mean squared error* (MSE) e o seu valor é encontrado a partir do quadrado do valor da diferença entre o valor predito e o valor real;
b) Erro Médio Absoluto: também conhecido como *mean absolute error* (MAE), é dado pela diferença absoluta entre os valores preditos e os valores reais;
c) Erro Médio Absoluto Percentual: chamado de *mean absolute percentage error* (MAPE), é bastante parecido com a métrica anterior, com a diferença de que o valor

[1] CHOLLET, François et al. Keras documentation.Disponível em: https://keras. io/. Acesso em: 15 ago. 2022

obtido pela subtração entre os dois parâmetros é dividido pelo valor real e multiplicado por 100, a fim de ser transformado em um valor percentual;

d) Coeficiente de determinação: parâmetro estatístico muito importante que define o ajuste do modelo gerado aos dados e quão bem o modelo gerado consegue explicar os dados. Assume valores de 0 a 1, sendo que quanto mais próximo de 1, melhor o modelo desenvolvido. Aparece também sob os termos R2 ou R^2, além de também possuir o termo em inglês R^2- *score.*

2.5 Python

Amplamente difundida atualmente, a linguagem *Python* é bastante utilizada nas áreas de Inteligência Artificial e Ciência de Dados, tanto para a manipulação de grandes conjuntos de dados, como também para o desenvolvimento de algoritmos e *frameworks* com funções especializadas para a montagem de programas de Aprendizado de Máquinas, em especial as Redes Neurais Artificiais.

A linguagem surgiu na Holanda no início dos anos 1990 criada por Guido Van Rossum para ser sucessora de uma já existente linguagem chamada ABC (VANROSSUM; DRAKE, 2010). Ao surgir com sua versão 1.0 nos anos 90, a linguagem Python foi sendo incrementada ao longo dos anos e tendo suas principais funções sendo criadas e incorporadas em sua estrutura. Pode ser considerada uma linguagem de alto nível, com vários pacotes de desenvolvimento que visam facilitar sua utilização e economizar o tempo dos desenvolvedores. De acordo com Chun (2001), algumas outras características da linguagem devem ser enaltecidas tais como sua escalabilidade, orientação a objetos, facilidade de leitura e escrita do código, portabilidade e robustez. Além disso, ainda de acordo com Chun (2001), a linguagem Python é uma linguagem interpretada, o que retira o tempo de compilação do desenvolvimento.

Todas essas vantagens unidas ao fato de a linguagem possuir uma vasta comunidade de apoio e documentação fez com que muitas empresas tomassem posse de sua utilização e criassem seus próprios *frameworks* a serem aplicados nas mais diferentes áreas. Por esse motivo, existem hoje bibliotecas poderosas, ainda que de fácil utilização, para a construção de Redes Neurais Artificiais que foram utilizadas neste trabalho.

2.5.1 *Tensor Flow* e *Keras*

Um forte exemplo de que grandes empresas veem na linguagem Python uma oportunidade de desenvolvimento de grandes *frameworks* é o *TensorFlow*[2]*,* uma complexa biblioteca voltada para computação numérica distribuída. Sua utilização é difundida através de vários projetos de Aprendizagem de Máquinas pela possibilidade de criar e executar Redes Neurais Artificiais tão complexas quanto necessário de forma eficiente. Foi criada pela *Google* e foi disponibilizada como código aberto em 2015 (GERÓN, 2019).

O *framework TensorFlow* além de disponibilizar várias bibliotecas especializadas, como bibliotecas de otimização, gráficas, focadas em Processamento de Linguagem Natural entre outras, também possui ferramentas de desenvolvimento que podem ser utilizadas em conjunto, como o *Google Colab.* O chamado *Google Colaboratory* é uma ferramenta que permite o desenvolvimento de código em conjunto e que é executado totalmente na Nuvem. Suas vantagens residem no fato de não ser necessária nenhuma configuração prévia, bem como no fácil e gratuito acesso a GPUs e também na facilidade de execução dos códigos escritos, separados e organizados em células de execução. Os códigos para este trabalho foram desenvolvidos utilizando essa ferramenta.

Relacionada ao *TensorFlow,* tem-se o surgimento também do *Keras,* uma API de Aprendizado Profundo de alto nível cuja implementação é feita através do próprio *TensorFlow.* Seu principal objetivo também é permitir a construção de Redes Neurais Profundas de forma rápida e intuitiva, através de funções prontas que podem ser carregadas e utilizadas em diversos conjuntos de dados.

2.5.2 *Streamlit*

Um dos *frameworks* mais utilizados na linguagem *Python* para a implantação de aplicações *web* é o *Streamlit*[3]. Com esse recurso, é possível integrar todo o código desenvolvido em *Python* em uma interface *web* para visualização rápida e fácil dos dados processados pelo algoritmo, facilitando a apresentação de gráficos e possibilitando uma interação intuitiva entre a aplicação e o usuário, que pode facilmente fornecer novos valores para determinados parâmetros a serem recalculados e mostrados. Essa tecnologia, além de possibilitar o desenvolvimento dessa interface, é também quem hospeda a aplicação. Neste

[2] ABADI, Martín et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.Disponível em: https://www.tensorflow.org/. Acesso em: 15 ago. 2022
[3] Streamlit documentation.Disponível em: https://streamlit.io/. Acesso em: 15 nov. 2022

trabalho, através do *Streamlit* foi desenvolvido um ambiente de simulação para o sensor virtual desenvolvido.

2.5.3 *Z-score*

Ainda estando diretamente ligado à utilização da linguagem *Python* e seus *frameworks*, é necessário que se faça uma breve explicação sobre uma determinada técnica estatística aplicada aos dados como um pré-processamento anterior ao desenvolvimento da Rede Neural. Nas bases de dados reais, algumas amostras de dados se distanciam de tal forma dos outros que são classificados como *outliers,* isto é, dados anômalos e que não devem ser considerados em análises posteriores. Uma das técnicas computacionais mais utilizadas para a remoção destes dados é a chamada pontuação Z, ou *Z-score*[4]. Essa métrica é calculada para cada amostra e mostra quantos desvios padrões ela está acima ou abaixo em relação à média populacional, ou o quão longe este valor está da média.

2.6 Kairos

Existem no mercado diversas opções de Sistemas Gerenciadores de Banco de Dados, onde cada qual possui as suas características de funcionamento próprias e sua própria estrutura de armazenamento. Os bancos de dados relacionais têm por característica marcante a sua estruturação em tabelas, onde cada registro é inserido em uma tabela e possui um identificador único. Por outro lado, os bancos não relacionais fogem dessa estrutura tradicional e propõem outras maneiras de armazenamento. Um desses bancos não relacionais é o *Apache Cassandra*[5], que possui código aberto e é utilizado por várias empresas devido à sua escalabilidade sem comprometimento de performance. Unido a este sistema está o *Kairos,* cujo código foi escrito unido ao *Cassandra* e cujo armazenamento dos dados é feito no próprio Cassandra. No entanto, o *Kairos* surge como um banco de dados específico de séries temporais, permitindo que sejam armazenados, consultados, agregados e visualizados diferentes pontos de dados ao longo de um determinado período de tempo. Para este trabalho, faz-se a utilização do *Kairos* para a coleta dos dados que foram armazenados no sistema referentes aos medidores instalados na caldeira siderúrgica que está sob análise.

---

[4] GLEN, Stephanie. Z-Score: Definition, Formula and Calculation.Disponível em: https://www.statisticshowto.com/probability-and-statistics/z-score/. Acesso em: 7 nov. 2022

[5] Apache Cassandra documentation. Disponível em: https://cassandra.apache.org/_/index.html. Acesso em: 15 ago. 2022

2.7 Sistema PIMS

A coleta e armazenamento de dados em um ambiente industrial é de vital importância para o conhecimento dos processos produtivos envolvidos. Nesse contexto, alguns sistemas existentes visam proporcionar a possibilidade de agregar esses dados em informação útil e de fácil acesso para os engenheiros de processo. Esses sistemas são conhecidos como sistemas PIMS (*Plant Information Management System)*.

De acordo com Carvalho et al. (2013) esses sistemas podem ser entendidos como sistemas que recuperam dados de diversas fontes distintas e os armazenam em um único banco de dados, onde podem ser visualizados através de ferramentas diversas. Existem vários benefícios atrelados à utilização desse tipo de sistema como a centralização dos dados, maior interatividade e possibilidade de geração de históricos de dados.

Em geral, a infraestrutura principal presente para sistemas PIMS consiste em um servidor principal onde os dados são centralizados. Estes dados são coletados do chão de fábrica via protocolo OPC (*Open Platform Communication*) e então enviados para o servidor central (CARVALHO et al., 2013).

Para este trabalho, é importante o entendimento destes conceitos pois foram utilizados na coleta dos dados no ambiente de produção e posterior envio ao banco de dados *Kairos.*

2.8 Protocolo OPC HDA

Por último, é necessária a explicação do protocolo de comunicação utilizado no mecanismo de coleta dos dados industriais. Como mostrado por Bochenek, Fojcik e Cupek (2011), o protocolo OPC é utilizado em diferentes sistemas para permitir acesso a dados para diferentes propósitos. De acordo com González et al. (2019), o protocolo OPC é uma interface de comunicação industrial que possibilita a interoperabilidade e heterogeneidade em aplicações de automação, já que permite a troca de dados entre o chão de fábrica e sistemas de controle mesmo que haja diferentes protocolos de comunicação entre a fonte e destino destes dados.

No entanto, apesar do protocolo OPC fornecer a possibilidade de troca de informações entre diferentes dispositivos com diferentes protocolos de comunicação, este trabalho utiliza na coleta de dados uma especificação diferente do OPC clássico, conhecido como OPC HDA (*historical data access*) que é uma definição dessa interface que permite o acesso a dados

históricos que posteriormente são enviados para o banco de dados de séries temporais responsável pelo armazenamento da planta siderúrgica em questão, o *Kairos.*

2.9 Trabalhos anteriores

Neste trabalho será utilizado uma Rede Neural Artificial para a estimativa do modelo de sensor virtual e, nesse sentido, na tentativa de reforçar a aplicabilidade desse tipo de algoritmo para esta finalidade, serão apresentados alguns exemplos de trabalhos anteriores que se utilizaram da mesma abordagem.

De uma forma parecida com o desenvolvido neste trabalho, em Shakil et al. (2009) um sensor virtual foi desenvolvido com a utilização de Redes Neurais dinâmicas para a medição da emissão de NO_X e O_2 durante o processo de combustão em um processo industrial.

Ko e Shang (2011) mostram um trabalho onde foi desenvolvido um sensor virtual baseado na utilização de uma Rede Neural especialista em análise de imagens para analisar a distribuição do tamanho de partículas. A principal ideia era possibilitar um modelo preditivo que pudesse ser usado em tempo real para operações na indústria.

Outro exemplo de aplicação é mostrado por Rani, Singh e Gupta (2013) , onde é realizada a construção de um sensor virtual para o controle de uma coluna de destilação. Neste caso, foi utilizado uma Rede Neural do tipo ADALINE para o controle de um processo de destilação.

Em Wang et al. (2019), foram utilizadas duas Redes Neurais Artificiais do tipo Convolucional para o desenvolvimento de um sensor virtual de aplicação e simulação em um caso para indústria química. As duas Redes foram combinadas no sentido de habilitar a manipulação de grandes bases de dados de processamento e manter a complexidade do modelo baixa.

Xibilia et al. (2020) mostra uma aplicação mais voltada para a segurança de um ambiente industrial, onde é feito o monitoramento de gases perigosos na planta através de um sensor virtual. Este sensor foi construído utilizando-se de duas abordagens a fim de um efeito de comparação entre os algoritmos utilizados: Análise de Componentes Principais (PCA) e uma Rede Neural Profunda.

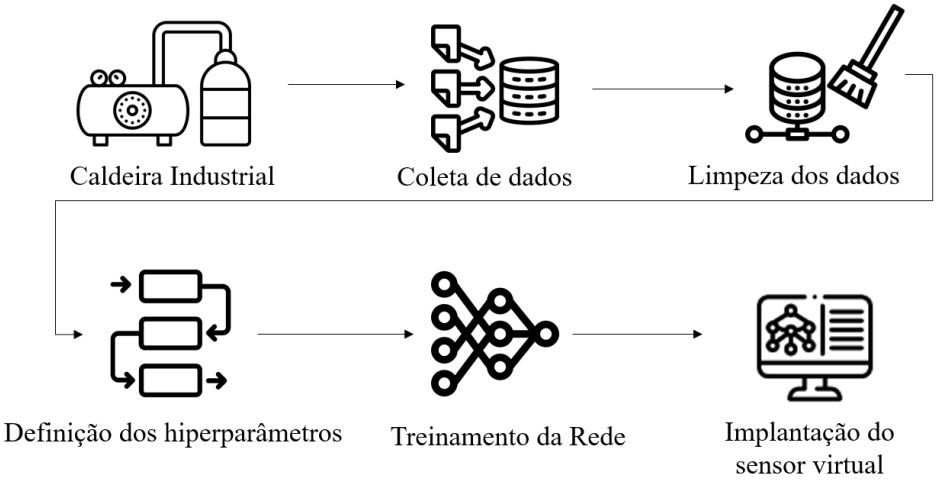
Desse modo, é possível observar como muitos trabalhos se aproveitam da generalização obtida por um modelo baseado em Redes Neurais Artificiais para o desenvolvimento de um sensor virtual que pode ser aplicado em diversas áreas, mas que é especialmente utilizado em ambientes industriais.

3 Materiais e Métodos

Neste capítulo serão apresentadas todas as técnicas utilizadas durante esta pesquisa, bem como os equipamentos usados para seu desenvolvimento, de modo a possibilitar a sua repetição caso desejado. Será explicada a princípio a parte referente a coleta e processamento dos dados, seguida pela seção referente ao treinamento da Rede Neural Artificial.

A Figura 3.1 mostra um fluxograma geral de todo o método implementado, passando pela coleta e processamento dos dados e chegando ao treinamento da Rede e implantação do sensor virtual. O treinamento da Rede será abordado de forma mais detalhada ao longo do capítulo mas, de uma forma geral, foi feito com os dados históricos vindos do sensor da caldeira e através de várias combinações dos parâmetros internos do algoritmo.

Figura 3.1 - Fluxograma geral do processo.



Fonte: Do autor (2022)

3.1 *Hardware* e *Software*

Para o desenvolvimento e treinamento da Rede Neural, bem como para a simulação do sensor virtual, foi utilizado um computador *notebook* com as seguintes características principais:

a) Sistema Operacional *Windows* 10 - 64 *bits;*
b) Processador Intel Core i5-5200 2.20 GHz (CPUs);
c) Memória de 6144 MB de RAM.

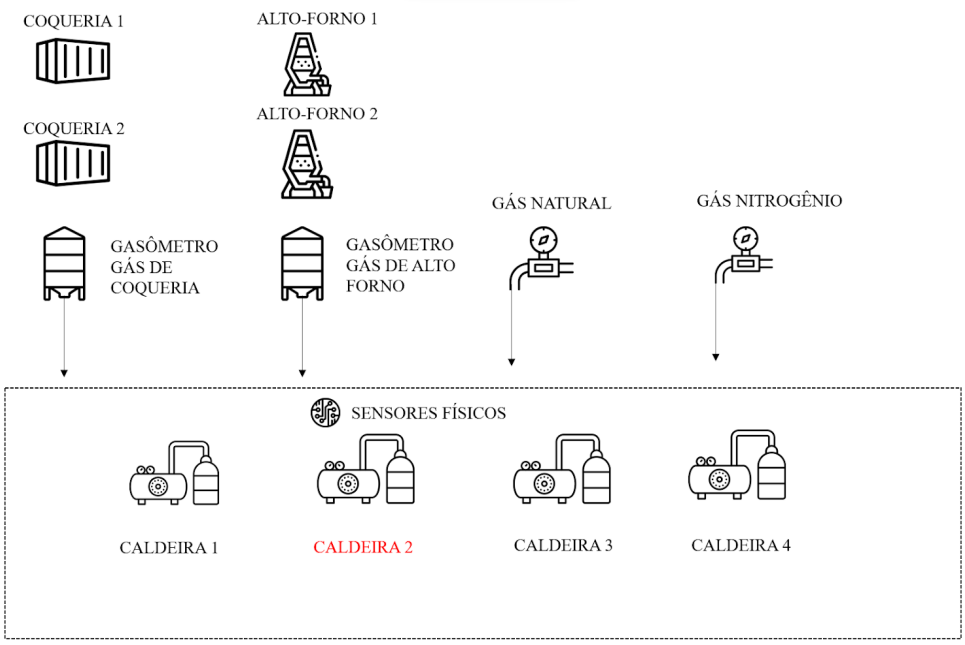
A análise dos dados, correções e construção da Rede Neural Artificial para estimativa do valor de saída foram realizados através da linguagem Python e seus *frameworks,* em

especial o *Keras.* Os dados foram obtidos do banco de dados de séries temporais da siderurgia em questão, o *Kairos.* O desenvolvimento do código foi feito basicamente através do *Google Colab,* cuja versão do *Python* utilizada foi a 3.7.15.

3.2 Coleta de dados

Para a realização deste trabalho, foi utilizada uma usina siderúrgica real, mais especificamente dados oriundos dos equipamentos instalados em sua Central Termoelétrica. Essa Central tem por principais objetivos a geração de vapor, energia elétrica e envio de ar soprado para os Altos-Fornos e possui 4 caldeiras com capacidade de 140 t/h cada. Além disso, os principais combustíveis utilizados são o Gás de Alto Forno, o Gás de Coqueria, o Gás de Aciaria e o Gás Natural, além do Gás Nitrogênio. A Figura 3.2 mostra uma visão geral dessa siderúrgica, onde a Caldeira 2 foi a utilizada para a realização deste trabalho.

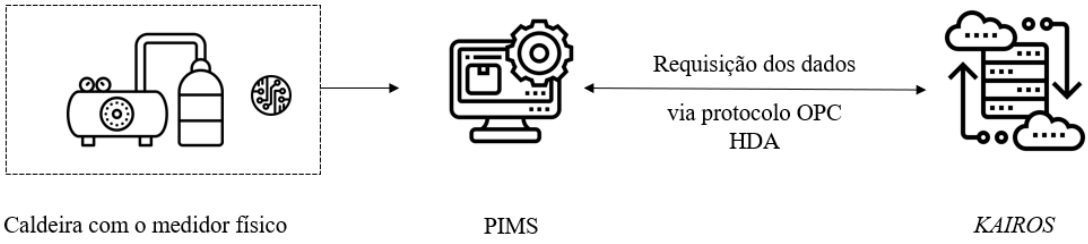
Figura 3.2 - Visão geral da Central Termoelétrica.



Fonte: Adaptado de Moreira (2017)

Os dados foram acessados através da ferramenta *Kairos,* banco de dados de séries temporais responsável unicamente por armazenar os dados históricos que foram coletados anteriormente do ambiente industrial. De uma forma simplificada, a Figura 3.3 ilustra o caminho realizado pelos dados até serem armazenados no *Kairos.*

Figura 3.3 - Envio dos dados ao *Kairos.*



Fonte: Do autor (2022)

O primeiro bloco do processo que envia os dados ao *Kairos* é composto pela caldeira industrial em si, bem como pelo sensor físico instalado na mesma, responsável por coletar as medições. Esses dados são enviados ao PIMS do sistema siderúrgico e salvos com um código identificador que indica que estes dados pertencem ao medidor da grandeza estudada. Os dados posteriormente são enviados ao *Kairos* depois de terem sido coletados do PIMS via protocolo OPC HDA e salvos com o código identificador deste medidor.

Os dados enviados ao *Kairos* após coletados da planta siderúrgica provêm de um medidor com resolução do tipo Minuto, isto é, com a capacidade de fornecer uma nova medição a cada minuto passado. Além disso, é um medidor do tipo periódico, que não acumula os valores ao longo do tempo, mas que a cada minuto envia um novo valor. Sua precisão é de 0,005% e, por oferecer uma medida de vazão, em toneladas por hora, a sua forma de agregação dos dados no sistema se dá através da média dos mesmos.

Na Figura 3.4 é possível visualizar o comportamento dos valores de gás de alto forno medidos entre os dias 1 e 8 de agosto de 2022, valores que depois foram exportados para um arquivo. Do mesmo modo, as Figura 3.5, 3.6 e 3.7 mostram o comportamento dos combustíveis que alimentaram a caldeira em questão neste mesmo intervalo de tempo, representando os valores de Gás de Coqueria, Gás Natural e Gás Nitrogênio, respectivamente. Por último, a Figura 3.8 ilustra o comportamento da variável de interesse a ser substituída posteriormente pelo sensor virtual, isto é, a quantidade de vapor gerada.

Figura 3.4 - Variação do gás de alto forno entre os dias 1 e 8 de agosto de 2022.

Fonte: Do Autor (2022)

Figura 3.5 - Variação do gás de coqueria entre os dias 1 e 8 de agosto de 2022.

Fonte: Do Autor (2022)

Figura 3.6 - Variação do gás natural entre os dias 1 e 8 de agosto de 2022.

Fonte: Do Autor (2022)

Figura 3.7 - Variação do gás nitrogênio entre os dias 1 e 8 de agosto de 2022.

Fonte: Do Autor (2022)

Figura 3.8 - Variação da quantidade de vapor entre os dias 1 e 8 de agosto de 2022.

Fonte: Do Autor (2022)

Para os combustíveis analisados, que estão na forma gasosa, a unidade de medida adotada é a Normal Metro Cúbico Por Hora, Nm^3/h. A geração de vapor é medida em Tonelada Por Hora.

3.3 Processamento dos dados

Na planta siderúrgica em questão, os medidores instalados para monitorar a Caldeira Industrial possuem uma resolução do tipo Minuto. Ao longo de 1 dia inteiro, existem cerca de 1440 pontos de dados para um único medidor. Como o período analisado corresponde a 7 dias, tem-se algo em torno de 10080 pontos de dados a serem analisados. Os dados foram salvos no formato *csv* e tinham o seguinte padrão:

a) Coluna 1: *Datetime,* isto é, coluna contendo as informações de data e hora nas quais o ponto foi coletado através do medidor físico instalado;
b) Coluna 2: *Raw Value*, que é o valor bruto medido pelo medidor, sem nenhum processamento adicional;

c) Coluna 3: *Cleansed Value*, coluna padrão que vem para os medidores sendo que alguns deles possuem um tipo de processamento, como tratamento de dados faltantes. Não é o caso para estes medidores;
d) Coluna 4: *Standardized Value,* responsável por alocar os valores padronizados ou normalizados dentro de algum critério. Alguns medidores possuem esses valores definidos, porém, para os medidores aqui utilizados, não há diferença entre os valores *raw, cleansed* e *standardized.*

Todas as 3 últimas colunas possuíam anexados aos cabeçalhos um texto conhecido como *metric code.* Este texto nada mais é do que a chave única pela qual o *Kairos* armazena e separa os medidores.

Além disso, por estarem no formato *csv,* por padrão era necessário que houvesse um delimitador separando estas colunas. No caso, pelo modo padrão com o qual o *Kairos* salva seus dados neste formato, o delimitador utilizado foi o ponto e vírgula.

Desse modo, primeiramente foi necessário utilizar a linguagem *Python* para um pré-processamento e organização dos dados. A princípio, cada conjunto foi lido e separado pelo delimitador padrão explicitado anteriormente, dando origem a uma massa de dados da forma ilustrada pela Figura 3.9.

Figura 3.9 - Primeiros valores da base de dados de Gás de Alto Forno após o primeiro processamento.

DateTime	MET00000000000031902460 (tag: point_type=cleansed)	MET00000000000031902460 (tag: point_type=raw)	MET00000000000031902460 (tag: point_type=standardized)
2022-08-01 03:00:00	34715,83595275879	34715,83595275879	34715,83595275879
2022-08-01 03:01:00	34876,13245646159	34876,13245646159	34876,13245646159
2022-08-01 03:02:00	36025,0483194987	36025,0483194987	36025,0483194987

Fonte: Do Autor (2022)

Na sequência, dado que os três tipos de dados das 3 últimas colunas eram iguais, a terceira e quarta colunas foram removidas das bases de dados. A fim de facilitar a leitura, o nome dos cabeçalhos também foi trocado, como mostrado na Figura 3.10.

Figura 3.10 - Visualização das 5 primeiras linhas para a base de gás de alto forno.

	DateTime	GAF
0	2022-08-01 03:00:00	34715,83595275879
1	2022-08-01 03:01:00	34876,13245646159
2	2022-08-01 03:02:00	36025,0483194987
3	2022-08-01 03:03:00	36175,23810068766
4	2022-08-01 03:04:00	35304,36064402262

Fonte: Do Autor (2022)

O mesmo foi feito para as bases de dados dos outros combustíveis, bem como da quantidade de vapor, com a diferença de que a coluna *Datetime* também foi removida para este, já que, pelo fato dos dados terem sido coletados no mesmo instante, a coluna mantida pela primeira base de dados serve também de referência para a última. Na sequência, todas as bases foram concatenadas entre si para a geração da base única, conforme a Figura 3.11.

Figura 3.11 - Bases concatenadas.

	DateTime	GAF	GCO	GN	N2	CONSUMO_ESP	VAPOR
0	2022-08-01 03:00:00	34715,83595275879	5187,429377110799	0	75,56529669594659	717,8234181548152	67,75316149393717
1	2022-08-01 03:01:00	34876,13245646159	5148,22630996704	0	75,04090741309308	715,0856064071916	67,9550340016683
2	2022-08-01 03:02:00	36025,0483194987	5126,148559188843	0	75,19088814315197	730,1080262301484	67,64256032307942
3	2022-08-01 03:03:00	36175,23810068766	5148,457555071513	0	75,9174204501861	724,742284489203	68,4329106648763
4	2022-08-01 03:04:00	35304,36064402262	5276,008960723876	0	75,69969747210514	718,5882149250173	68,83295885721843

Fonte: Do Autor (2022)

Alguns outros tratamentos foram realizados a fim de alimentar a Rede Neural com a maior qualidade de dados possível. Após a concatenação das bases, foi realizada uma análise estatística auxiliar a fim de verificar se haveria dados que fugissem do padrão, isto é, *outliers*. Para o treinamento de algoritmos de Inteligência Artificial, é interessante que estes valores sejam tratados a fim de generalizar o modelo da melhor forma possível. Existem várias maneiras de detectar *outliers* entre os dados, cada qual com suas peculiaridades. Neste trabalho, foi calculado o chamado *Z-score* (pontuação Z) para cada um dos valores. O valor limite para o *Z-score* a partir do qual os dados foram considerados anômalos foi 3.Após a aplicação do teste, houve uma redução de 10081 para 9659, de modo que 422 valores foram retirados por serem considerados *outliers*.

Na sequência, após a exclusão dos valores ditos como anômalos, a base de dados foi dividida em dados para treinamento do algoritmo e posteriormente teste da performance do mesmo. Mais especificamente, os conjuntos auxiliares que surgiram são os mostrados abaixo:

a) Conjunto de treinamento: a base de dados foi dividida na proporção de 70% de dados de treinamento e 30% de dados para teste, logo este conjunto representa a maior massa de dados entre os gerados após a divisão. São todos os dados de entrada que alimentaram a Rede Neural durante o seu processamento de treinamento. É formado especificamente pelos dados das vazões de combustível que alimentam a caldeira e que alimentam também a Rede Neural para seu treinamento. Unidos a esses dados também estão os valores de saída, utilizados durante o treinamento para fazer o mapeamento entre os valores de entrada e saída;
b) Conjunto de teste: corresponde a 30% da base original e também é formada pelas vazões de combustível. Agora, porém, estes são os valores que são fornecidos à Rede treinada para gerar os valores correspondentes de vapor, isto é, os valores de saída oriundos das entradas pelos vetores de pesos encontrados anteriormente.

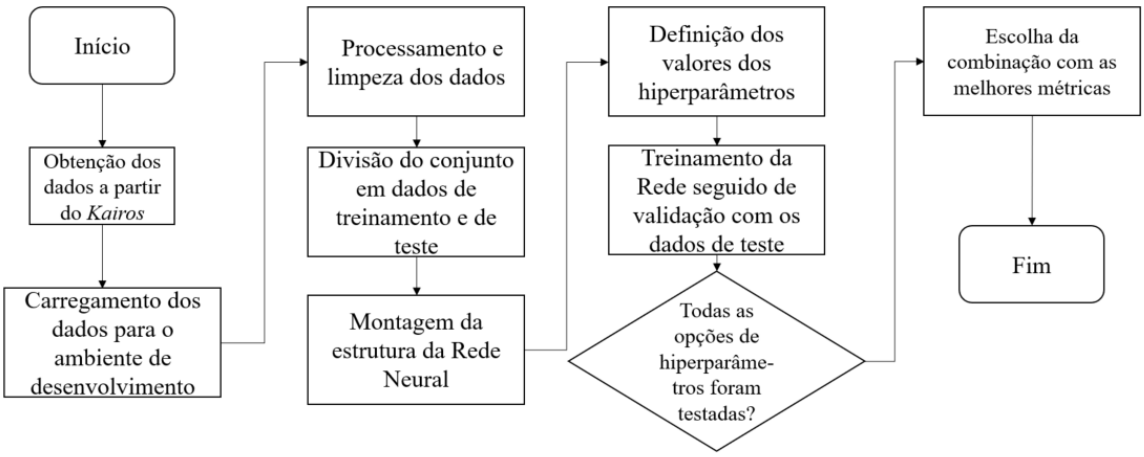
Estes foram os dados utilizados para o treinamento da Rede Neural e para estimar os valores de vapor gerados, a partir das vazões de combustível.

3.4 Treinamento da Rede Neural

O processo de treinamento da Rede foi apenas uma etapa do processo de desenvolvimento do algoritmo. Como demonstrado na seção anterior, foi necessário que os dados, após terem sido carregados do *Kairos* passassem por uma etapa de pré-processamento e limpeza a fim de permitir resultados melhores durante o treinamento e validação do algoritmo.

De uma forma geral, o desenvolvimento resumido pode ser explicado pela Figura 3.12, que sucintamente mostra as etapas desde o carregamento dos dados até o desenvolvimento do algoritmo em si.

Figura 3.12 - Fluxograma das etapas de desenvolvimento do código.



Fonte: Do Autor (2022)

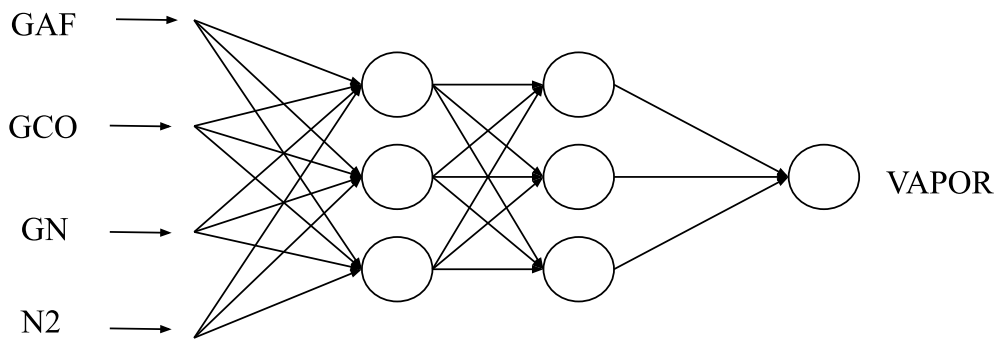
Observa-se que a parte inicial do desenvolvimento fez referência principalmente aos dados e sua organização, para que depois houvesse um direcionamento maior para a parte da codificação do algoritmo. Como essa é a parte principal do desenvolvimento, faz-se necessária a explicação de como foi estruturada a Rede Neural. A princípio, o seu esqueleto foi montado seguindo a estrutura:

- Neurônios de entrada: Levando em conta que a base de dados de treinamento possui 4 previsores, representando as vazões de Gás de Alto Forno, Gás de Coqueria, Gás Natural e Gás Nitrogênio, então a quantidade de neurônios de entradas foi configurada para acompanhar esse valor de previsores.
- Camadas ocultas: O número de camadas ocultas em uma Rede Neural está diretamente ligado à complexidade do problema e à quantidade de processamentos que os dados devem sofrer até atingir a última camada. Para este caso, 2 camadas ocultas obtiveram bons resultados de treinamento. Em cada camada oculta, a função de ativação escolhida foi a função *reLU*.
- Neurônios em cada camada oculta: O número de neurônios nas camadas ocultas pode ser bastante variável e, em geral, é escolhido com uma abordagem de tentativa e erro. Alguns mecanismos oferecem um auxílio para um bom palpite, como a soma do número de neurônios de entrada com o número de neurônios na saída, dividido por 2. Para este trabalho foi escolhido o valor 3.

- Neurônio de saída: Os problemas de regressão requerem um único neurônio na saída, composto pelo valor inferido. A função de ativação para este último neurônio deve ser a função linear.

Desse modo, a estrutura da Rede fica da seguinte forma mostrada na Figura 3.13.

Figura 3.13 - Estrutura da Rede Neural montada.



Fonte: Do Autor (2022)

Uma outra parte importante durante o treinamento das Redes Neurais é a definição de seus outros hiperparâmetros como o tamanho do lote, o número de épocas, a função de perda, as métricas de avaliação e os otimizadores.

Em relação às métricas de avaliação do modelo, é importante observar que para análises de regressão, não é medida a acurácia propriamente dita do modelo, mas sim valores de erro que, quanto menores forem, melhor avaliado será o modelo. Unindo-se às métricas utilizadas, valores diferentes de hiperparâmetros foram submetidos a um treinamento que buscou a sua melhor combinação possível. Desse modo, foram passados para a Rede Neural os valores que deveriam ser assumidos por estes hiperparâmetros e a avaliação feita através das métricas foi responsável por indicar a melhor combinação dentre todas.

Os valores assumidos pelos hiperparâmetros foram:

a) Número de épocas: a faixa de valores assumidos pelo número de épocas a princípio foi 100, 200 e 300 épocas. Essa variação é importante pois, apesar de ser necessário um número significativo de épocas para um bom treinamento, este valor não deve ser exagerado para que não haja desperdício de poder computacional em um algoritmo que já atingiu o mínimo de sua função de erro;

b) Tamanho do lote: os valores possíveis passados à Rede foram 50, 100 e 150, correspondendo ao número de registros que seriam processados antes da atualização dos pesos;
c) Otimizadores: diretamente relacionado à taxa de aprendizagem do algoritmo, foram passados 2 possíveis otimizadores disponíveis no *Keras* para a Rede: *Adam* e SGD;
d) Inicialização dos pesos: dado que o treinamento consiste justamente nos ajustes dos valores de peso, esse parâmetro indica como será feita a inicialização de seus valores. Para este caso de treinamento foi passado somente o tipo de inicialização aleatório.

A partir destas combinações foi possível a obtenção do melhor modelo possível de Rede Neural. A medida de desempenho do algoritmo foi realizada através da previsão a partir do conjunto de testes, onde as métricas de Erro Absoluto Médio (MAE), Erro Absoluto Médio Percentual (MAPE) e Coeficiente de Determinação (R^2) foram coletadas e apresentadas no formato tabular no capítulo seguinte.

4 Resultados e Discussão

Nesta seção serão apresentados os resultados pertinentes oriundos da modelagem e treinamento do algoritmo apresentado no capítulo anterior.

A partir das combinações possíveis de hiperparâmetros formadores da Rede Neural usada, foi calculado para cada uma delas o Erro Absoluto Médio (MAE) e Erro Absoluto Médio Percentual (MAPE) para o conjunto de treinamento. Em seguida, essas mesmas métricas foram calculadas para o conjunto de teste bem como o Coeficiente de Determinação. A Tabela 1 reúne a combinação de hiperparâmetros com as respectivas métricas obtidas.

Tabela 1 - Combinação de parâmetros e valores de métricas.

Épocas	**Lote**	**Otimizador**	**MAE (treino)**	**MAPE (treino)**	**MAE (teste)**	R^2 **(teste)**	**MAPE (teste)**
300	50	*Adam*	2,25	2,79	2,13	0,813	0,026
300	100	*Adam*	2,23	2,76	2,16	0,809	0,026
300	150	*Adam*	4,56	5,60	7,01	0,052	0,085
200	50	*Adam*	2,19	2,70	2,39	0,787	0,029
200	100	*Adam*	2,40	2,97	2,86	0,739	0,035
200	150	*Adam*	2,21	2,75	2,16	0,816	0,026
100	50	*Adam*	2,20	2,73	2,13	0,814	0,026
100	100	*Adam*	2,52	3,12	2,33	0,794	0,029
100	150	*Adam*	4,39	5,39	3,41	0,675	0,041

Fonte: Do Autor (2022)

A combinação de parâmetros também levou em conta o otimizador conhecido como *SGD,* também parte do *Keras.* Em comparação com o otimizador *Adam,* o *SGD* possui uma taxa de aprendizagem cerca de 10 vezes maior, fazendo com que a descida do gradiente seja mais abrupta. Os resultados obtidos com esse otimizador não foram satisfatórios, de modo que somente o *Adam* foi utilizado.

Para as combinações realizadas foi possível observar uma influência positiva da diminuição do tamanho do lote para o coeficiente de determinação do modelo. Uma outra observação a ser feita é em relação ao modo de inicialização dos pesos. Como isso se dá de forma aleatória, em alguns casos ocorreram valores de erro mais acentuado, devido a uma inicialização ruim dos pesos. Logo, pela tabela acima foi possível a escolha da melhor combinação de parâmetros. Além das próprias métricas obtidas, foi observado que um menor tamanho do lote e um maior número de épocas forneceu uma maior segurança de melhores resultados, tendo uma performance melhor para os casos onde a inicialização aleatória dos pesos não foi favorável. A combinação escolhida é mostrada na Tabela 2.

Tabela 2 - Parâmetros com melhor resultado.

Número de Épocas	300
Tamanho do lote	50
Otimizador	*Adam*
Pesos	Aleatório

Fonte: Do Autor (2022)

Os vetores de pesos foram aleatoriamente inicializados com os valores referenciados pela Tabela 3, que indica os valores para cada uma das camadas. Os valores de *bias* foram omitidos pois se iniciaram zerados.

Tabela 3 - Pesos iniciais.

-0,0800	0,5543	0,4093
0,2885	-0,5892	-0,0324
-0,6425	0,2886	0,3509
0,1551	-0,6276	-0,8602
0,5596	-0,6330	-0,3194
0,4019	-0,4827	-0,6843
-0,4141	-0,0088	0,9220
-0,0168	0,4909	-0,1290

Fonte: Do Autor (2022)

Durante o treinamento, foi percebido que não era necessário um grande número de épocas como pensado a princípio mas que com apenas um pequeno número, em torno de 15 épocas, o algoritmo já convergia para valores baixos de erro. Desse modo, ao final do treinamento os valores ajustados para os pesos são mostrados na tabela para as respectivas camadas e também os valores dos *bias* são mostradas na Tabela 4, Tabela 5, Tabela 6, Tabela 7 e Tabela 8.

Tabela 4 - Pesos finais entre as entradas e primeira camada oculta.

-0,0016	0,5673	0,3712
0,4709	-0,5249	-0,1143
0,6154	0,2230	0,1205
-0,0160	-0,6660	-0,8513

Fonte: Do Autor (2022)

Tabela 5 - *Bias* para os neurônios da primeira camada oculta.

2,2790	1,8669	-1,8727

Fonte: Do Autor (2022)

Tabela 6 - Pesos finais entre a primeira e segunda camadas ocultas.

0,7716	-0,6330	-0,4299
0,3443	-0,4827	-0,7262
-0,4179	-0,0088	0,8894

Fonte: Do Autor (2022)

Tabela 7 - *Bias* para os neurônios da segunda camada oculta.

2,1678	0	-0,0403

Fonte: Do Autor (2022)

Tabela 8 - Pesos finais entre a última camada e a saída e *bias* de saída.

-0,0282
0,1074
0,1372
2,3308

Fonte: Do Autor (2022)

Dessa forma, foi obtido um vetor com os valores previstos a partir dos dados de teste. A partir desse vetor, as métricas finais obtidas para esta melhor combinação são mostradas na Tabela 9.

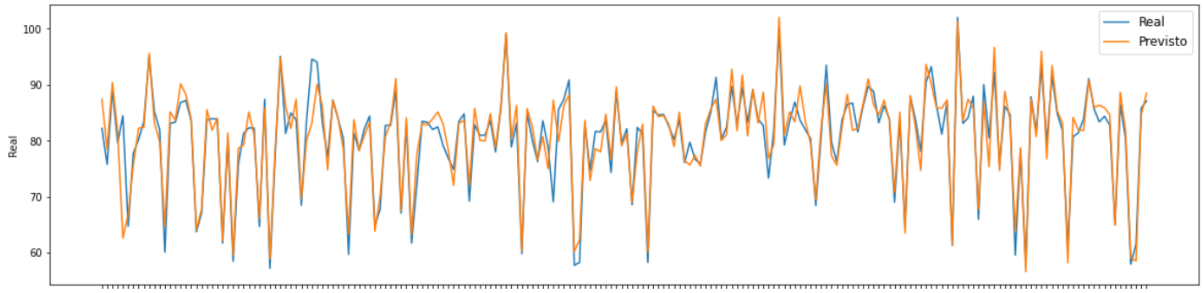
Tabela 9 - Valores finais para as métricas.

Erro Médio Absoluto (MAE)	2,13
Erro Médio Absoluto Percentual (MAPE)	2,63 %
R^2	0,813

Fonte: Do Autor (2022)

Para efeito de comparação entre os valores previstos e os valores reais que haviam sido guardados para comparação, foi criada uma base de dados auxiliar formada pelos dois vetores de valores. Além disso, uma coluna extra formada por valores de data e hora foi concatenada a fim de permitir que os dados fossem apresentados a partir do formato de série temporal. Os 200 primeiros valores previstos pela Rede Neural e comparados com os valores reais são mostrados na Figura 4.1.

Figura 4.1 - Comparação entre valores previstos e reais.



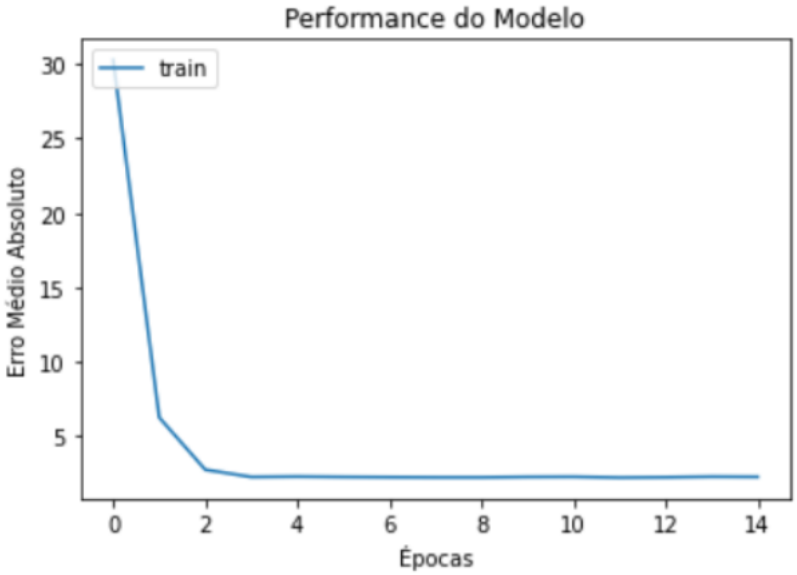
Fonte: Do Autor (2022)

Ao longo do tempo, observa-se como os valores previstos estiveram sempre aproximados aos valores reais de vapor gerados, dadas as quantidades de combustível apresentadas ao modelo.

Pela observação do gráfico, é possível observar a tendência do modelo de prever valores superestimados. Isso significa que os erros observados se dão pelo fato do modelo estimar valores acima dos valores reais.

Além disso, uma outra visualização possível de ser observada na Figura 4.2 com o treinamento do modelo, é a queda no valor do Erro Médio Absoluto a partir do aumento na quantidade de épocas de treinamento. Tanto a função de perda, isto é, a função utilizada durante o treinamento da rede onde o ajuste dos pesos foi feito de tal modo a minimizá-la, como a métrica utilizada para avaliação utilizaram como base esse tipo de erro. A diminuição do seu valor ao longo do treinamento mostra um bom ajuste dos dados e um mapeamento eficiente entre as entradas e saída do algoritmo.

Figura 4.2 - Gráfico Épocas *versus* Erro Médio Absoluto.



Fonte: Do Autor (2022)

4.1 Implantação do Sistema

A fim de possibilitar testes mais aprofundados e simulação do funcionamento do sensor baseado no algoritmo treinado, foi desenvolvida uma aplicação de implantação do modelo.

Para o desenvolvimento da aplicação foi utilizado um computador *notebook*, com as seguintes características principais:

a) Sistema Operacional *Windows* 10 - 64 *bits;*
b) Processador Intel Core i5-5200 2.20 GHz (CPUs);
c) Memória de 6144 MB de RAM.

O código foi desenvolvido com a linguagem *Python* na versão 3.7.15 através da ferramenta *Sublime Text 3* em união ao *framework* conhecido como *Streamlit.* Após o desenvolvimento, a aplicação é hospedada pelo próprio serviço oferecido pelo *framework*, podendo ser acessada a qualquer pessoa em posse do *link.*

A aplicação que hospeda o sensor é dividida em 3 grandes páginas. A primeira delas é uma página inicial com informações básicas sobre o projeto, contando com uma breve explicação sobre a proposta, os dados e o algoritmo. Na sequência, existe uma página com explicações detalhadas sobre a base de dados utilizada. Nela, é possível observar os dados

dos combustíveis, e entender como a Rede Neural foi configurada. Por último, a parte mais importante da aplicação se encontra na página de previsão e testes.

A previsão da quantidade de vapor gerada para a Central Termoelétrica dessa siderúrgica, como foi dito, é de extrema importância para o acompanhamento dos processos que dependem deste recurso. Por esse motivo, a aplicação oferece dois modelos de teste e previsões. O primeiro deles é a opção de fornecer ao sensor virtual, valores para os combustíveis separadamente, através de interfaces amigáveis ao usuário, como mostrada na Figura 4.3. Após a seleção, é feito o processamento destes valores e um valor esperado de vapor é fornecido ao usuário.

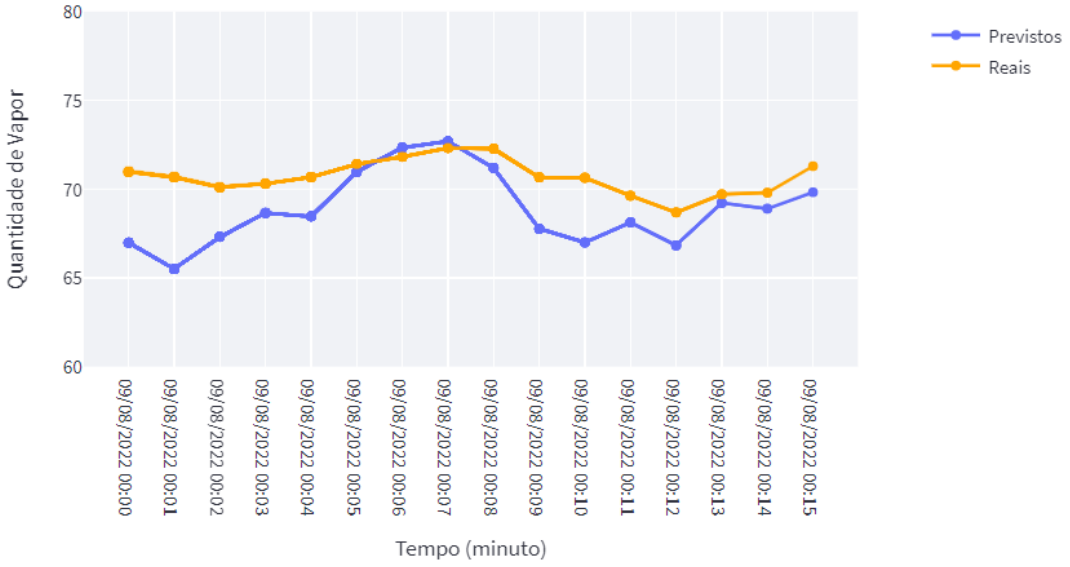
Figura 4.3 - Interface desenvolvida para previsão de valor único.

Fonte: Do Autor (2022)

No entanto, a principal funcionalidade deste sensor virtual é o fornecimento de informações em tempo contínuo e na mesma resolução que seria oferecida pelo sensor físico. Isso significa, que a partir dos valores presentes de combustíveis, a cada minuto uma nova informação de vapor deve ser mostrada ao usuário. Para que isso ficasse de uma forma fácil

de acompanhar, foi desenvolvido para o sistema, um gráfico de pontos que é atualizado automaticamente à medida que novos pontos de vapor vão sendo estimados. Para isso, é necessário fazer o *upload* de um conjunto de dados formado pelas vazões de combustível, para que seja simulada a entrada de dados fornecida pelos sensores físicos. A partir desses valores, o sensor virtual formado pelo algoritmo treinado disponibiliza, a cada minuto, uma nova informação de vapor. A Figura 4.4 mostra o gráfico atualizado após alguns instantes.

Figura 4.4 - Gráfico de comparação dos valores em tempo real.



Fonte: Do Autor (2022)

Desse modo, surgem duas grandes possibilidades para a utilização do sensor virtual desenvolvido. A primeira é a sua utilização conjunta com o sensor físico. Como foi explicitado, muitas vezes a falha na obtenção dos dados pode ser tão grave quanto a parada completa de funcionamento do sensor. Isso pois, o levantamento de informações que não refletem de fato os valores de produção impacta diretamente no funcionamento dos processos dependentes e muitas vezes demora a ser descoberto, o que pode causar danos graves à produção. Neste caso, é interessante que se faça uma comparação entre o erro existente entre o valor previsto e real, e caso haja uma discrepância acentuada entre ambos por um certo período de tempo, um alarme é ativado solicitando a análise e reparo do sensor físico.

O outro caso é o funcionamento deste sensor como substituto do físico em casos de avaria ou manutenção do mesmo. Como apresentado, os resultados obtidos pela rede foram satisfatórios, não se afastando dos valores reais em valores acentuados. Para uma siderúrgica

com uma Central Termoelétrica com processos interligados como esta, a manutenção do monitoramento dessa variável é de grande interesse.

5 Conclusão

O objetivo geral proposto pelo trabalho foi o desenvolvimento de um sensor virtual para estimar a quantidade de vapor gerada por uma caldeira através de Redes Neurais Artificiais. Com isso, seria possível viabilizar a substituição do medidor físico dessa grandeza ou fornecer a possibilidade do uso simultâneo do dispositivo físico e virtual a fim de comparar os valores obtidos e aumentar a segurança dos processos da Central Termoelétrica.

Através de variadas configurações de hiperparâmetros para a Rede Neural, onde diversas combinações possíveis foram testadas tanto para os dados de treino, onde a métrica principal foi o Erro Médio Absoluto (MAE), bem como para os dados de teste onde unindo-se ao MAE foram calculadas também o coeficiente de determinação R^2 e o Erro Médio Absoluto Percentual (MAPE), obteve-se os valores satisfatórios de 2,13, 0,813 e 2,63%, mostrando uma boa capacidade de generalização unida a um baixo erro percentual de previsão.

A implantação do modelo no ambiente de testes e simulação, que possibilitou a comparação minuto a minuto entre os dados previstos e reais, deixou claro que o desenvolvimento de um sensor virtual para esta aplicação é uma ideia promissora, confirmando que a utilização das Redes Neurais Artificiais se mostrou uma boa escolha como algoritmo desenvolvedor do sensor virtual. Sua utilização pode ser de extrema importância para esta Central Termoelétrica, permitindo a obtenção de valores em caso de falha do medidor físico e de possíveis comparações entre os valores reais e previstos a fim de serem determinadas métricas de calibração e qualidade dos dados.

5.1 Propostas de continuidade

Como propostas de continuidade para trabalhos futuros surgem ideias como:

a) Utilização de diferentes algoritmos de Aprendizagem de Máquinas para o desenvolvimento do sensor virtual e comparação dos resultados com as Redes Neurais;
b) Utilização de uma base de dados com mais amostras, abrangendo um período de coleta que supere o intervalo de 1 semana utilizado neste trabalho;
c) Utilização de outros *frameworks* e linguagens de programação a fim de proporcionar uma abordagem estatística mais robusta durante a limpeza e processamento de dados, como a linguagem R.

REFERÊNCIAS

A. K. Jain, Jianchang Mao,K. M. Mohiuddin.Artificial neural networks: a tutorial. **Computer**, vol. 29, no. 3, pp. 31-44, Mar.1996.

ARBIB, Michael A. **Brain theory and neural networks**.Cambridge: MIT Press, MA, 1995. 1255 p.

BISHOP, Christopher M.; NASRABADI, Nasser M. **Pattern recognition and machine learning**. New York: Springer, 2006. 482 p.

BOCHENEK, Michał; FOJCIK, Marcin; CUPEK, Rafał. OPC Historical Data Access–OPC Foundation Toolkit Improvement Suggestions. **International Conference on Computer Networks**. Springer, Berlin, Heidelberg, 2011. p. 338-347. DOI: 10.1007/978-3-642-21771-5_37. Disponível em: https://link.springer.com/chapter/10.1007/978-3-642-21771-5_37. Acesso em: 16 nov.2022.

BOTELHO, Manoel Henrique Campos; BIFANO, Hercules Marcello. **Operação de caldeiras: gerenciamento, controle e manutenção**. 2.ed.São Paulo:Editora Blucher, 2015. 209 p.

BROWNLEE, JASON. **What is the Difference Between a Batch and an Epoch in a Neural Network?,** 2022. Disponível em: https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/ . Acesso em: 15 ago.2022

CARVALHO, Fabio Barros de et al. Sistemas PIMS-conceituação, usos e benefícios. **Tecnologia em Metalurgia, Materiais e Mineração**, v. 1, n. 4, p. 1-5, 2013. DOI:10.4322/tmm.00104001. Disponível em: https://tecnologiammm.com.br/article/10.4322/tmm.00104001/pdf/1573492069-1-4-1.pdf. Acesso em 16 nov.2022.

CHUN, Wesley J. **Core Python Programming.** Upper Sadle River: Prentice Hall Professional, 2001**.** 771 p.

COSTA, Fernando Corner Da. Leia sobre: Gases combustíveis siderúrgicos. **Portal Aquecimento Industrial**, Campinas, 2019. Disponível em: https://www.aquecimentoindustrial.com.br/saiba-mais-sobre-a-coluna-gases-combustiveis-siderurgicos/#:~:text=Sua%20composi%C3%A7%C3%A3o%20volum%C3%A9trica%20m%C3%A9dia%20%C3%A9,%2C0%20%E2%80%93%207%2C8%20%25. Acesso em: 15 jul.2022.

CSP limpa os gases de aciaria sem consumo de água. **ABM - Associação Brasileira de Metalurgia, Materiais e Mineração.** Disponível em: https://www.abmbrasil.com.br/por/noticia/csp-limpa-os-gases-de-aciaria-sem-consumo-de-agua. Acesso em: 16 jul.2022

D.A. Tillman, A.J. Rossi, W.D. Kitto, **Wood Combustion, Principles, Processes, and Economics**.New York:Academic Press, 1981. 205 p.

DE OLIVEIRA FONSECA, Marcos. Comunicação OPC–Uma abordagem prática.**VI Seminário de Automação de Processo,** 2002.

DEEP Learning Book. **Data Science Academy,** 2022. Disponível em: https://www.deeplearningbook.com.br/o-neuronio-biologico-e-matematico. Acesso em: 10 jul.2022

ELKELAWY, Medhat,ALM EDIN, Hagar. **Boilers and Steam Generation**, 2019. Disponível em: https://www.researchgate.net/publication/331411946_Boilers_and_Steam_Generation. Acesso: 15 jul. 2022.

FERREIRA, Emerson Lamartine. **Análise do Comportamento da Combustão do Gás do Alto Forno-BFG em uma Câmara de Combustão de Turbina a Gás**.2015.Tese (Mestrado em Conversão de Energia) - Universidade Federal de Itajubá, Itajubá, 2015.

FORTUNA, Luigi et al. **Soft sensors for monitoring and control of industrial processes**. London, UK: Springer,,2007. 267 p.

GÉRON, Aurélien. **Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems**. 3. ed.Sebastopol:O'Reilly Media, Inc., 2019. 850 p.

GONZALEZ, G. D. Soft sensors for processing plants. **Proceedings of the Second International Conference on Intelligent Processing and Manufacturing of Materials. IPMM 99 (Cat. No. 99 EX 296).** IEEE, p. 59-69, 1999.

GONZÁLEZ, Isaías et al. A literature survey on open platform communications (OPC) applied to advanced industrial environments. **Electronics**, v. 8, n. 5, p. 510, 2019. DOI: https://doi.org/10.3390/electronics8050510. Disponível em: https://www.mdpi.com/2079-9292/8/5/510. Acesso em: 16 nov.2022.

HAYKIN, Simon. **Redes neurais:** princípios e prática. 2.ed.Porto Alegre: Bookman Editora, 2007. *E-book.* 866 p.

KO, Young-Don; SHANG, Helen. A neural network-based soft sensor for particle size distribution using image analysis. **Powder Technology**, v. 212, n. 2, p. 359-366, 2011. DOI:https://doi.org/10.1016/j.powtec.2011.06.013. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S0032591011002968. Acesso em: 13 jul.2022.

LAWN, Christopher J. **Principles of combustion engineering for boilers**. Orlando:Academic Press, 1987. 628 p.

LOPES, R. P. et al. Controle da combustão em fornalhas a lenha. **Proceedings of the 3. Encontro de Energia no Meio Rural**, Campinas, 2000. Disponível em: http://www.proceedings.scielo.br/scielo.php?pid=MSC0000000022000000200023&script=sci_arttext&tlng=pt. Acesso em: 15.jul.2022.

LOTUFO, Francisco Antonio; GARCIA, Claudio. Sensores virtuais ou soft sensors: Uma introdução. **Proc. of the 7th Brazilian Conference on Dynamics, Control and Applications,** Presidente Prudente. p. 1-9, 2008. Disponível em: https://www.researchgate.net/profile/Francisco-Lotufo-2/publication/228413086_Sensores_Virtuais_ou_Soft_Sensors_Uma_introducao/links/53fb90fd0cf2dca8fffe7f21/Sensores-Virtuais-ou-Soft-Sensors-Uma-introducao.pdf. Acesso em: 12 jul.2022.

LUTTMANN, Reiner et al. Soft sensors in bioprocessing: a status report and recommendations. **Biotechnology journal**, v. 7, n. 8, p. 1040-1048, 2012. DOI:

https://doi.org/10.1002/biot.201100506. Disponível em: https://onlinelibrary.wiley.com/doi/full/10.1002/biot.201100506. Acesso em: 13 jul.2022.

MCALLISTER, Sara; CHEN, Jyh-Yuan; FERNANDEZ-PELLO, A. Carlos. **Fundamentals of combustion processes**. New York: Springer, 2011. 328 p.

MCCULLOCH, Warren S.; PITTS, Walter. A logical calculus of the ideas immanent in nervous activity. **The bulletin of mathematical biophysics**, v. 5, n. 4, p. 115-133, 1943.

MENGHINI, D. et al. Effect of excess air on the optimization of heating appliances for biomass combustion. **Experimental Thermal and Fluid Science**, v. 32, n. 7, p. 1371-1380, 2008. DOI: https://doi.org/10.1016/j.expthermflusci.2007.11.018. Disponível em: https://www.sciencedirect.com/science/article/pii/S0894177707001707. Acesso em: 13 jul.2022.

MOREIRA, José Geraldo da. **Metodologia para determinação contínua do consumo específico por combustível e priorização da distribuição de combustíveis entre caldeiras**. Tese (Mestrado em Engenharia da Energia) - Universidade Federal de São João Del Rei, São João Del Rei, 2017.

RAGLAND, Kenneth W.; BRYDEN, Kenneth M.; KONG, Song-Charng. **Combustion engineering**. Boca Raton, FL: CRC press, 2011. 599 p.

RANI, Asha; SINGH, Vijander; GUPTA, J. R. P. Development of soft sensor for neural network based control of distillation column. **ISA transactions**, v. 52, n. 3, p. 438-449, 2013. DOI:https://doi.org/10.1016/j.isatra.2012.12.009. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S0019057812002030. Acesso em: 13 jul.2022.

RAO, Valluru; RAO, Hayagriva V. **C++ neural networks and fuzzy logic**. Mis: Press, 1995. 454 p.

ROTHMAN, Denis. **Artificial intelligence by example: develop machine intelligence from scratch using real artificial intelligence use cases**.Mumbai:Packt Publishing Ltd,2018. 453 p.

SHAKIL, M. et al. Soft sensor for NOx and O2 using dynamic neural networks. **Computers & Electrical Engineering**, v. 35, n. 4, p. 578-586, 2009. DOI: https://doi.org/10.1016/j.compeleceng.2008.08.007. Disponível em: https://www.sciencedirect.com/science/article/pii/S0045790608000876. Acesso em: 13 jul.2022.

SHARMA, Sagar; SHARMA, Simone; ATHAIYA, Anidhya. Activation functions in neural networks. **IJEAST**, vol. 4, n. 12, p. 310-316, 2020. Disponível em: https://www.ijeast.com/papers/310-316,Tesma412,IJEAST.pdf. Acesso em :10 jul.2022

SILVA, Ivan Nunes da; SPATTI, Danilo Hernane; FLAUZINO, Rogério Andrade. **Redes neurais artificiais para engenharia e ciências aplicadas**.2.ed.São Paulo: Artliber Editora Ltda, 2010. 429 p.

TURNS, Stephen R. **Introdução à Combustão:** Conceitos e Aplicações.3.ed. AMGH Editora, 2013. 393 p.

VANDAGRIFF, Ralph. **Practical guide to industrial boiler systems**.Boca Raton: CRC Press,2001. 362 p.

VANROSSUM, Guido; DRAKE, Fred L. **The python language reference**. Amsterdam, Netherlands: Python Software Foundation, 2010. 155 p.

WANG, Kangcheng et al. Dynamic soft sensor development based on convolutional neural networks. **Industrial & Engineering Chemistry Research**, v. 58, n. 26, p. 11521-11531, 2019.DOI: 10.1021/acs.iecr.9b02513.Disponível em: https://pubs.acs.org/doi/abs/10.1021/acs.iecr.9b02513. Acesso em: 12 jul.2022.

WOOD, Thomas. Softmax Function. Deepai.org. Disponível em: https://www.deepai.org/machine-learning-glossary-and-terms/softmax-layer. Acesso em: 11 jul.2022.

XIBILIA, Maria Gabriella et al. Soft sensors based on deep neural networks for applications in security and safety. **IEEE Transactions on Instrumentation and Measurement**, v. 69, n. 10, p. 7869-7876, 2020. DOI: https://doi.org/10.1109/TIM.2020.2984465. Disponível em: https://ieeexplore.ieee.org/abstract/document/9058770. Acesso em 13 jul.2022.

YEGNANARAYANA, Bayya. **Artificial neural networks**. New Delhi: PHI Learning Pvt Ltd.,2009. 461 p.

APÊNDICE A - Parte do *script* em *python*

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import datetime
from scipy import stats
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error as mae
from sklearn.metrics import mean_absolute_percentage_error
from sklearn.metrics import r2_score
from keras.models import Sequential
from keras.layers import Dense

dadosGAF = pd.read_csv("GAF.csv", delimiter = ";")
dadosGCO = pd.read_csv("GCO.csv", delimiter = ";")
dadosGN = pd.read_csv("GN.csv", delimiter = ";")
dadosN2 = pd.read_csv("N2.csv", delimiter = ";")
dadosVAPOR = pd.read_csv("Vapor.csv", delimiter = ";")

dadosGAF.drop('MET000000000031902460<br>(tag: point_type=raw)', inplace=True, axis=1)
dadosGAF.drop('MET000000000031902460<br>(tag: point_type=standardized)', inplace=True,
axis=1)
dadosGAF.columns = ['DateTime', 'GAF']


dadosGCO.drop('MET000000000033042877<br>(tag: point_type=raw)', inplace=True, axis=1)
dadosGCO.drop('MET000000000033042877<br>(tag: point_type=standardized)', inplace=True,
axis=1)
dadosGCO.drop('DateTime', inplace=True, axis=1)
dadosGCO.columns = ['GCO']


dadosGN.drop('MET000000000031902444<br>(tag: point_type=raw)', inplace=True, axis=1)
dadosGN.drop('MET000000000031902444<br>(tag: point_type=standardized)', inplace=True,
axis=1)
dadosGN.drop('DateTime', inplace=True, axis=1)
dadosGN.columns = ['GN']


dadosN2.drop('MET000000000130874165<br>(tag: point_type=raw)', inplace=True, axis=1)
dadosN2.drop('MET000000000130874165<br>(tag: point_type=standardized)', inplace=True,
axis=1)
dadosN2.drop('DateTime', inplace=True, axis=1)
dadosN2.columns = ['N2']
```

```
dadosVAPOR.drop('MET0000000000031902552<br>(tag: point_type=raw)', inplace=True, axis=1)
dadosVAPOR.drop('MET0000000000031902552<br>(tag: point_type=standardized)', inplace=True,
axis=1)
dadosVAPOR.drop('DateTime', inplace=True, axis=1)
dadosVAPOR.columns = ['VAPOR']
dadosVAPOR.head()

dadosConsumoEsp = pd.read_csv("ConsumoEsp.csv", delimiter = ";")

dadosConsumoEsp.drop('MET0000000000037313670<br>(tag: point_type=raw)', inplace=True,
axis=1)
dadosConsumoEsp.drop('MET0000000000037313670<br>(tag: point_type=standardized)',
inplace=True, axis=1)
dadosConsumoEsp.drop('DateTime', inplace=True, axis=1)
dadosConsumoEsp.columns = ['CONSUMO_ESP']

dados = pd.concat([dadosGAF, dadosGCO, dadosGN, dadosN2, dadosConsumoEsp, dadosVAPOR],
axis=1)

dadosSemConsumo = dados.drop(['CONSUMO_ESP'], axis = 1)
dadosSemConsumo.head()
dadosSemConsumo = dadosSemConsumo.apply(lambda x: x.str.replace(',','.'))
dadosSemConsumo['GAF'].head()

# Conversão dos tipos de dados
dadosSemConsumo['GAF']=dadosSemConsumo['GAF'].astype(float)
dadosSemConsumo['GCO']=dadosSemConsumo['GCO'].astype(float)
dadosSemConsumo['GN']=dadosSemConsumo['GN'].astype(float)
dadosSemConsumo['N2']=dadosSemConsumo['N2'].astype(float)
dadosSemConsumo['VAPOR']=dadosSemConsumo['VAPOR'].astype(float)

# Correlações
cor = dadosSemConsumo.corr()
f, ax = plt.subplots(figsize = (10,10))
sns.heatmap(cor, annot = True)

dadosSemData = dadosSemConsumo.drop(['DateTime'], axis = 1)

z = np.abs(stats.zscore(dadosSemData))
threshold = 3

dados_sem_outliers = dadosSemData[(z < 3).all(axis=1)]
previsores = dados_sem_outliers.iloc[:,0:4].values
vapor = dados_sem_outliers.iloc[:,4].values

X_train, X_test, y_train, y_test = train_test_split(previsores, vapor, test_size = 0.3, random_state =
42)
```

```
def compareFunction (epocs, batches, optimizers, weights, train, target, prev_test, target_test):
    for wgt in weights:
        for ep in epocs:
            for bt in batches:
                for opt in optimizers:
                    print("calculating for kernel = " + str(wgt) + " and epochs = " + str(ep) + " and bt = " +
str(bt) + " and opt = " + str(opt))
                    print("\n")
                    regressor.add(Dense(units = 3, activation = 'relu', kernel_initializer = wgt ,input_dim =
4))
                    regressor.add(Dense(units = 3, activation = 'relu', kernel_initializer = wgt))
                    regressor.add(Dense(units = 1, activation = 'linear'))
                    regressor.compile(loss = 'mean_absolute_error', optimizer = opt, metrics =
['mean_absolute_error'])
                    regressor.fit(train,target, batch_size = bt, epochs = ep, verbose = 0)
                    previsoes = regressor.predict(prev_test)
                    MAE = mae(target_test, previsoes)
                    MAPE =  mean_absolute_percentage_error(target_test, previsoes)
                    R2 = r2_score(target_test, previsoes)
                    print("MAE: " + str(MAE))
                    print(" MAPE: " + str(MAPE))
                    print(" r2: " + str(R2))


weights = ['random_uniform', 'normal']
optimizers = ['adam', 'sgd']
batches = [50,100,150]
epocs = [100, 200, 300]
testFunction(epocs, batches, optimizers, weights, X_train, y_train, X_test, y_test)


regressor = Sequential()
regressor.add(Dense(units = 3, activation = 'relu', input_dim = 4))
regressor.add(Dense(units = 3, activation = 'relu'))
regressor.add(Dense(units = 1, activation = 'linear'))
regressor.compile(loss = 'mean_absolute_error', optimizer = 'adam', metrics =
['mean_absolute_error'])


history = regressor.fit(X_train,y_train, batch_size = 50, epochs = 15)
previsoes = regressor.predict(X_test)
previsoes = pd.DataFrame(previsoes, columns = ['Previsão'])

real = pd.DataFrame(y_test, columns = ['Real'])
df  = pd.concat([previsoes,real], axis=1)
dados = dadosSemConsumo
data = dados.iloc[:,0].values
data = pd.DataFrame(data, columns = ['Data'])
```

```
df  = pd.concat([previsoes,real, data], axis=1)
df_aux = df.iloc[0:200,:]

plt.figure(figsize = (20,5))

sns.lineplot(data = df_aux, x = "Data", y = "Real")
sns.lineplot(data = df_aux, x = "Data", y = "Previsão")
plt.legend(labels=["Real","Previsto"],fontsize = 'large')


plt.plot(history.history['mean_absolute_error'])
plt.title('Performance do Modelo')
plt.ylabel('Erro Médio Absoluto')
plt.xlabel('Épocas')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

regressor_json = regressor.to_json()
with open('regressor.json','w') as json_file:
  json_file.write(regressor_json)
regressor.save_weights('regressor.h
```

APÊNDICE B - *Script* aplicação *web*

Script utilizado para o desenvolvimento da aplicação *web* hospedada pelo *Streamlit:*

```
import streamlit as st
from PIL import Image
from keras.models import model_from_json
import numpy as np
import json
import pandas as pd
import time
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
from itertools import count
import streamlit.components.v1 as components
import time
import plotly.express as px
from plotly.subplots import make_subplots
import plotly.graph_objects as go

st.set_page_config(page_title=None, page_icon=None, layout='centered',
initial_sidebar_state='expanded')

def pag_inicial():

        st.title("Soft Sensor - Redes Neurais Artificiais")
        st.markdown("---")
        st.markdown("### Pedro Oliveira Annoni Farah - Trabalho de Conclusão de Curso  ")
        st.markdown("---")
        st.markdown("Esta aplicação foi desenvolvida para que fosse possível implantar e testar um
modelo de predição construído para se estimar o teor de vapor gerado a partir de uma reação de
`Combustão` em uma `Caldeira Industrial` de uma siderúrgica real. ")
        st.markdown("---")
        st.markdown("")
        st.markdown("### ⚡ Sobre a proposta")
        st.markdown("")

        st.info("A proposta do trabalho foi realizar o desenvolvimento de um Sensor Virtual, isto é,
um estimador de uma variável de interesse a partir de variáveis de entrada e que pudesse substituir um
sensor físico em uma planta siderúrgica  ")
        st.markdown("### 📊 Sobre os dados")
        st.markdown("")
        st.info("Os dados foram coletados a partir de um banco de dados de Séries Temporais de uma
siderúgica real entre os dias 1 e 8 de Agosto de 2022. Eles são oriundos de medidores instalados em
uma Caldeira Industrial específica da Central Termoelétrica desta siderúrgica. ")
        st.markdown("### 💻 Sobre o algoritmo")
        st.markdown("")
```

```
        st.info("Para o desenvolvimento do Sensor Virtual foi desenvolvido uma Rede Neural Artificial capaz de predizer um valor de saída de interesse, no caso, a quantidade de vapor gerada a partir das variáveis de entrada coletadas. ")

        st.markdown("<div align='center'><br>"
          "<img src='https://img.shields.io/badge/Feito%20COM-PYTHON%20-red?style=for-the-badge'"
          "alt='API stability' height='25'/>"
          "<img src='https://img.shields.io/badge/EDITADO%20COM-SUBLIME-blue?style=for-the-badge'"
          "alt='API stability' height='25'/>"
          "<img src='https://img.shields.io/badge/DESENVOLVIDO%20COM-Streamlit-green?style=for-the-badge'"
          "alt='API stability' height='25'/></div>", unsafe_allow_html=True)

        st.markdown("---")
add_sidebar = st.sidebar.selectbox("Opções de escolha: ", ('Início','Base','Previsão e testes'))
if(add_sidebar == 'Início'):
        pag_inicial()

if (add_sidebar == 'Base'):
        st.title("📑 Base")
        st.markdown("---")
        st.markdown("As variáveis de entrada utilizadas para previsão da variável de saída foram escolhidas com base em estudos relacionados ao processo de Combustão como um todo. Sabe-se que dois dos principais componentes de uma reação de Combustão são os `Combustíveis` e os `Comburentes`. O primeiro pode ser composto por diversas substâncias, podendo inclusive ser sólido, líquido ou gasoso. Enquanto isso, o comburente geralmente é o próprio ar atmosférico. Para o caso estudado, as variáveis de entrada utilizadas nessa caldeira são as seguintes: ")
        '''
        - [x]  Gás de Alto Forno - GAF (Nm3/h)
        - [x]  Gás de Coqueria - GCO (Nm3/h)
        - [x]  Gás Natural - GN (Nm3/h)
        - [x]  Gás Nitrogênio - N2 (Nm3/h)

        '''
        st.markdown("")
        st.warning("Juntamente com os combustíveis que alimentam a Caldeira está a quantidade de Ar fornecido como comburente para a reação. Caso a quantidade seja insuficiente, tem-se a Combustão Incompleta, onde nem toda a quantidade de combustível reage, levando à formação do tóxico Monóxido de Carbono. Caso a quantidade esteja em excesso, apesar de reagir com todo o combustível, esta quantidade adicional acaba por roubar calor dos produtos diminuindo a eficiência da reação.")
        st.markdown("")
        '''

        Abaixo é possível escolher uma das variáveis de entrada para visualizar o seu comportamento durante os dias 1 e 8 de agosto de 2022:
```

```
        '''
        option = st.selectbox("Qual variável gostaria de visualizar?",("GAF","GCO","GN","N2"))
        st.markdown("")
        if (option == "GAF"):
                image = Image.open('GAFIMG.png')
                st.image(image, caption='Gás de Alto Forno (Nm3/h)')
        if (option == "GCO"):
                image = Image.open('GCOIMG.png')
                st.image(image, caption='Gás de Coqueria (Nm3/h)')
        if (option == "GN"):
                image = Image.open('GNIMG.png')
                st.image(image, caption='Gás Natural (Nm3/h)')
        if (option == "N2"):
                image = Image.open('N2IMG.png')
                st.image(image, caption='Gás Nitrogênio (Nm3/h)')

        st.markdown("Um parâmetro interessante que pode ser visualizado é a correlação entre as variáveis de entrada e saída. Neste caso, a variável de saída está sendo considerada como a quantidade de vapor: ")
        image = Image.open('CORR.png')
        st.image(image, caption='Correlação entre variáveis')
        st.markdown("")
        st.markdown("")
        st.markdown("A princípio, a Rede Neural foi montada com os seguintes parâmetros: ")
        '''
        - [x]  2 camadas ocultas
        - [x]  3 neurônios em cada camada oculta
        - [x]  Função de Ativação 'reLU' entre as camadas ocultas
        - [x]  Otimizador 'Adam'
        - [x]  Função de Perda 'mean absolute error'
        - [x]  Métrica principal também 'mean absolute error'


        '''
        st.markdown("")
        st.markdown("Para um treinamento com 30 épocas e tamanho do Lote igual a 50, obteve-se os valores que podem ser comparados com os valores reais no gráfico abaixo:  ")
        image = Image.open('RESCOMP.png')
        st.image(image, caption='Comparação entre valores previstos e reais')

if (add_sidebar == 'Previsão e testes'):
        st.title(" 🔍 Previsão e testes!")
        st.markdown("---")
        '''
        O funcionamento do teste é bastante simples e intuitivo. É necessário que você informe os valores para as seguintes variáveis:

 1. Fluxo de Gás de Alto Forno
```

2. Fluxo de Gás de Coqueria
3. Fluxo de Gás Natural
4. Fluxo de Gás Nitrogênio

```
        '''
        st.markdown("---")
        st.markdown("### Gás de Alto Forno")
        st.markdown("Primeiramente, é necessário fornecer a vazão de Gás de Alto Forno que estará alimentando a Caldeira Industrial nesse momento: ")
        GAF = st.number_input('Vazão de Gás de Alto Forno (Nm3/h)')
        st.markdown("### Gás de Coqueria")
        st.markdown("Do mesmo modo, também é necessário informar a vazão de Gás de Coqueria: ")
        GCO = st.number_input('Vazão de Gás de Coqueria (Nm3/h)')
        st.markdown("### Gás Natural")
        st.markdown("Escolha agora a vazão de Gás Natural através do slider:  ")
        GN = st.slider('Vazão de Gás Natural (Nm3/h)', 0,15000)
        st.markdown("### Gás Nitrogênio")
        st.markdown("Agora, escolher a vazão de Gás Nitrogênio através do slider:  ")
        N2 = st.slider('Vazão de Gás Nitrogênio (Nm3/h)', 0,15000)

        previsor = np.array([[GAF,GCO,GN,N2]])
        if st.button("Clique aqui para calcular a quantidade vapor esperada!"):
                arquivo = open('./Regressorr.json', 'r')
                estrutura = arquivo.read()
                arquivo.close()

                regressor = model_from_json(estrutura)
                regressor.load_weights('./Regressorr.h5')
                resultado = regressor.predict(previsor)
                st.info(resultado[0][0])

        carregar = st.selectbox("Carregar dados para prever saídas?: ", ('Sim','Não'))
        if carregar == "Sim":
                upload_file = st.file_uploader("Escolha um arquivo CSV", type = 'csv')
                if upload_file is not None:
                        data = pd.read_csv(upload_file)
                        st.write(data)
                        st.success("Dados importados com sucesso")
                        if st.button("Estimar valores..."):
                                resultados = []
                                x_vals = []
                                y_vals = []
                                y_vals2 = []
                                previsores = data.iloc[0:1440,1:5].values
                                st.write(previsores)
                                valorReal = data.iloc[0:1440,5].values;
                                arquivo = open('./Regressorr.json', 'r')
```

```
estrutura = arquivo.read()
arquivo.close()
regressor = model_from_json(estrutura)
regressor.load_weights('./Regressorr.h5')
previsoes = regressor.predict(previsores)
previsoes = pd.DataFrame(previsoes, columns = ['Previsão'])
st.write(previsoes)
valorReal = pd.DataFrame(valorReal, columns = ['Vapor'])
st.write(valorReal)
time1 = data.iloc[0:1440,0].values
time1 = pd.DataFrame(time1, columns = ['Data'])
st.write(time1)
df  = pd.concat([time1,previsoes], axis=1)
df2 = pd.concat([time1,valorReal], axis=1)
x = df['Data'].values
y = df['Previsão'].values
y2 = df2['Vapor'].values
index = count()
plt.style.use('fivethirtyeight')
fig = px.line()

subfig = make_subplots(specs=[[{"secondary_y":True}]])
the_plot = st.plotly_chart(subfig)
def animate (i):
        x_vals.append(x[i])
        y_vals.append(y[i])
        y_vals2.append(y2[i])

        plt.cla()
        plt.plot(x_vals,y_vals)
        plt.title("Valores estimados ao longo do tempo")
        plt.xlabel("Tempo")
        plt.ylabel("Quantidade de Vapor")
        if i == 0:
                subfig.add_trace(go.Scatter(x = x_vals, y = y_vals,
name = "Previstos",marker= {'color': '#636EFA'}))
                subfig.add_trace(go.Scatter(x = x_vals, y = y_vals2,
name = "Reais",marker= {'color': 'orange'}))
        else:
                subfig.add_trace(go.Scatter(x = x_vals, y = y_vals,
showlegend = False, marker= {'color': '#636EFA'}))
                subfig.add_trace(go.Scatter(x = x_vals, y = y_vals2,
showlegend = False,marker= {'color': 'orange'}))
        subfig.update_xaxes(title_text = "Tempo (minuto)")

        subfig.update_yaxes(title_text = "Quantidade de Vapor")
        subfig.update_traces(mode='markers+lines')
        subfig.update_layout(
```

```
            xaxis_title="Tempo (minuto)",
            yaxis_title="Quantidade de Vapor",
            yaxis_range=[60,80]
            )
            the_plot.plotly_chart(subfig)
for i in range(1000):
            animate(i)
            time.sleep(10)
```