



UNIVERSIDADE FEDERAL DE ALAGOAS  
UNIDADE ACADÊMICA CENTRO DE TECNOLOGIA  
CURSO DE ENGENHARIA QUÍMICA



ERICK CARVALHO DA SILVA

**TÉCNICAS DE *DEEP LEARNING*: *AUTOENCODER* E *LONG SHORT-TERM MEMORY (LSTM)* NO DESENVOLVIMENTO DE SENSORES VIRTUAIS E NA DETECÇÃO E DIAGNÓSTICO DE FALHAS EM PROCESSOS REACIONAIS COMPLEXOS**

Maceió - AL

2023

ERICK CARVALHO DA SILVA

**TÉCNICAS DE *DEEP LEARNING*: *AUTOENCODER* E *LONG SHORT-TERM MEMORY (LSTM)* NO DESENVOLVIMENTO DE SENSORES VIRTUAIS E NA DETECÇÃO E DIAGNÓSTICO DE FALHAS EM PROCESSOS REACIONAIS COMPLEXOS**

Trabalho de Conclusão de Curso apresentado ao curso de Engenharia Química da Universidade Federal de Alagoas como requisito parcial para obtenção do título de Bacharel em Engenharia Química.

Orientador: Prof. Dr. Frede de Oliveira Carvalho

Maceió - AL


2023



ERICK CARVALHO DA SILVA


**TÉCNICAS DE *DEEP LEARNING*: *AUTOENCODER* E *LONG-SHORT TERM MEMORY* (LSTM) NO DESENVOLVIMENTO DE SENSORES VIRTUAIS E NA DETECÇÃO E DIAGNÓSTICO DE FALHAS EM PROCESSOS REACIONAIS COMPLEXOS**

**BANCA EXAMINADORA**

Documento assinado digitalmente  
 FREDE DE OLIVEIRA CARVALHO  
Data: 29/05/2023 15:23:09-0300  
Verifique em <https://validar.iti.gov.br>


---

Prof. Dr. Frede de Oliveira Carvalho

Documento assinado digitalmente  
 JOAO INACIO SOLETTI  
Data: 30/05/2023 14:59:24-0300  
Verifique em <https://validar.iti.gov.br>

---

Prof. Dr. João Inácio Soletti

Documento assinado digitalmente  
 WILLIAM IMAMURA  
Data: 30/05/2023 16:02:38-0300  
Verifique em <https://validar.iti.gov.br>

---

Prof. Dr. William Imamura

Catálogo na fonte  
Universidade Federal de Alagoas  
Biblioteca Central  
Divisão de Tratamento Técnico  
Bibliotecário: Jone Sidney A. de Oliveira – CRB-4 –1485

S586t Silva, Erick Carvalho da

Técnicas de deep learning : autoencoder e long short-term memory (lstm) no desenvolvimento de sensores virtuais e na detecção e diagnóstico de falhas em processos reacionais complexos/ Erick Carvalho da Silva. - Maceió: AL, 2023.

63f. : il.

Orientador(a): Frede de Oliveira Carvalho

Monografia (Trabalho de Conclusão de Curso em Engenharia Química). Universidade Federal de Alagoas. Centro de Tecnologia. Maceió, 2021.

Inclui bibliografia: f.61-63.

1. Autoencoder. 2. Deep learning. 3. softsensor - LSTM.

I. Título.

CDU: 66.011

Dedico este trabalho a Deus que me deu forças, e à minha mãe que sempre acreditou nos sonhos do seu filho.

## **AGRADECIMENTOS**

Agradeço a Deus por sempre me dar condições de realizar meus sonhos e estar ao meu lado durante toda minha trajetória.

À minha mãe, Francinete, que nunca mediu esforços para que eu tivesse condições de alcançar meus objetivos, sempre acreditou no meu potencial e é a minha força motriz para continuar almejando novas conquistas. Ao meu pai, Heleno, que contribuiu, financeiramente, para a minha permanência em Alagoas.

À minha namorada, Lara, que me motivou com muito amor e carinho, me inspirou com sua dedicação e competência em todas as suas batalhas, e que compartilha todos os momentos da vida comigo.

Aos meus irmãos que Deus me enviou, Adenilson, Leon, Raquel e Vinicius, que sempre me apoiaram e estiveram presentes em todos os momentos, minha vida não seria a mesma sem a presença de vocês nela. Um agradecimento especial para Mayumi, Lara, Leon e Vinicius, que me acolheram em Alagoas e se tornaram parte da minha família.

Aos meus amigos de graduação, Gabrielle, Iury, Leon, Mário, Thalyta e Vinicius, compartilhamos juntos todos os momentos de felicidade e aflição ao longo dessa jornada. Ao meu amigo e companheiro de pesquisas, Mário, que me apoiou e me aconselhou durante o desenvolvimento deste trabalho. À Gisele, uma grande amiga que dividiu apartamento comigo durante toda a nossa jornada de graduação.

Aos professores da Universidade Federal de Alagoas, dos quais fui aluno, por terem contribuído com a minha formação.

Ao Programa de Educação Tutorial – Ciência e Tecnologia, ao qual tive o privilégio de ser membro, por ter contribuído na minha permanência no curso e ter me proporcionado ajudar no avanço da educação sendo um agente de transformação, e através do PET desfrutei das melhores experiências que a Universidade oferece, e que impactaram minha vida pessoal e profissional. Durante essa trajetória fiz amigos que levarei para vida. Agradeço aos meus tutores Eduardo Lucena e Luciano Barbosa e aos amigos Alana, Ewerton, Iany, Jadson, Mario, Rayssa, Valéria e Vinicius, foi muito gratificante compartilhar os momentos com vocês.

Ao meu orientador, Dr. Frede de Oliveira Carvalho, por ter prestado todo suporte necessário e ter tido cuidado em contribuir na garantia de um trabalho de qualidade, agradeço pela paciência e ensinamentos.

Por fim, agradeço a todos aqueles que contribuíram em minha vida e ajudaram a tornar esse sonho realidade.

“E, quanto fizerdes por palavras ou por obras, fazei tudo em nome do Senhor Jesus, dando por ele graças a Deus Pai.”

[Colossenses, 3:17]





## RESUMO

Com a modernização e aplicação da Inteligência Artificial no âmbito da indústria 4.0, as técnicas de *deep learning* têm sido amplamente abordadas com a alternativa de otimizar os processos industriais evitando erros gerados por falhas humanas ou por equipamentos físicos que não conseguem processar grandes quantidades de dados. Os maiores usos dessas técnicas de *deep learning*, atualmente, são em processos químicos, inferência estatística com sensores virtuais e detecção e diagnóstico de falhas. O presente trabalho tem por objetivo utilizar técnicas *deep learning* atuais, sendo elas *autoencoder* e *Long-Short-Term-Memory (LSTM)*, para desenvolver sensores virtuais e estratégias de detecção e diagnóstico de falhas em sistemas reacionais complexos. Os modelos foram criados em duas etapas, a primeira direcionada para a criação, treinamento e avaliação de sensores virtuais para diferentes conjuntos de dados, e a segunda etapa foi direcionada pra a criação de estratégias de detecção e diagnóstico de falhas em 2 cenários diferentes: *LSTM* na mesma etapa, e o segundo cenário utilizou *autoencoder* para detecção e *LSTM* para diagnóstico de falhas. Os cenários foram avaliados e comparados posteriormente. Os resultados obtidos revelaram que os sensores virtuais desenvolvidos com a técnica de *deep learning LSTM* apresentaram melhor desempenho utilizando dois conjuntos de dados: conjunto com 3 segundos de tempo de amostragem e 2 tempos de atraso, apresentando 0,999998 no coeficiente de determinação e  $0,37 \times 10^{-8}$  de erro quadrático médio; e o conjunto com 9 segundos de tempo de amostragem e 3 tempos de atraso, apresentando 0,999998 no coeficiente de determinação e  $0,35 \times 10^{-8}$  de erro quadrático médio. No que se refere à detecção e diagnóstico de falhas, a utilização conjunta das técnicas *Autoencoder* e *LSTM* no segundo cenário obtiveram resultados superiores com acurácia de 99% e 1,02% de taxa de erro, enquanto o cenário 1 apresentou 98% de acurácia e 1,18% de taxa de erro. Essa abordagem integrada do cenário 2 mostrou-se mais eficaz na identificação de falhas, destacando sua relevância prática no contexto de sistemas industriais complexos. Portanto, pode-se concluir que as técnicas de *deep learning*, analisadas neste estudo, fornecem soluções promissoras para a simulação de sensores virtuais e a detecção de falhas em sistemas reacionais complexos. Além disso, essas descobertas corroboram a importância do presente trabalho no desenvolvimento de novas tecnologias para a formação do engenheiro químico, especialmente diante dos desafios apresentados pela indústria 4.0.

**Palavras-Chave:** *Autoencoder*, *Deep learning*, sensor virtual, *LSTM*.

## ABSTRACT

With the modernization and application of Artificial Intelligence in the context of Industry 4.0, deep learning techniques have been extensively explored as an alternative to optimize industrial processes, mitigating errors caused by human failures or physical equipment limitations in processing large amounts of data. Currently, the major applications of deep learning techniques include chemical processes, statistical inference with softsensors, and fault detection and diagnosis. This study aims to utilize current deep learning techniques, namely Autoencoder and Long-Short-Term Memory (LSTM), to develop softsensor and fault detection and diagnosis strategies in complex reactive systems. The models were created in two stages: the first stage focused on the creation, training, and evaluation of softsensors for different datasets, while the second stage aimed at developing fault detection and diagnosis strategies in two distinct scenarios: LSTM in the same stage, and the second scenario utilized Autoencoder for fault detection and LSTM for fault diagnosis. The scenarios were subsequently evaluated and compared. The results obtained revealed that softsensors developed using LSTM deep learning technique exhibited better performance using two datasets: one with a sampling time of 3 seconds and 2 times delays, yielding a coefficient of determination of 0.999998 and a mean squared error of  $0.37 \times 10^{-8}$ ; and another dataset with a sampling time of 9 seconds and 3 time delays, presenting a coefficient of determination of 0.999998 and a mean squared error of  $0.35 \times 10^{-8}$ . Regarding fault detection and diagnosis, the combined use of Autoencoder and LSTM techniques in the second scenario achieved superior results with 99% accuracy and a 1.02% error rate, while scenario 1 showed 98% accuracy and a 1.18% error rate. This integrated approach in scenario 2 proved to be more effective in fault identification, highlighting its practical relevance in the context of complex industrial systems. Therefore, it can be concluded that the deep learning techniques analyzed in this study provide promising solutions for softsensor simulation and fault detection in complex reactive systems. Furthermore, these findings support the importance of this work in the development of new technologies for the education of chemical engineers, especially considering the challenges posed by Industry 4.0.

**Keywords:** Autoencoder, Deep learning, softsensor, LSTM.

## LISTA DE FIGURAS

Figura 1- Representação esquemática de um reator de van de Vusse .....	21
Figura 2 - Fontes de falhas em um processo .....	31
Figura 3 - Modelo de uma RNA.....	33
Figura 4 - Estrutura de RNA <i>LSTM</i> .....	35
Figura 5 - Estrutura de uma rede neural Autoencoder.....	37
Figura 6 - Fluxograma da 1º parte do trabalho.....	40
Figura 7 - Fluxograma da 2º parte do trabalho.....	40
Figura 8 - Cenário 1 para a detecção e diagnóstico de falhas.....	44
Figura 9 - Cenário 2 para a detecção e diagnóstico de falhas.....	45
Figura 10 - Matriz de confusão .....	47
Figura 11 - Gráficos de paridade dos sensores virtuais com melhores tempos de amostragem e tempos de atraso .....	52
Figura 12 - Matriz de confusão da FDD utilizando <i>LSTM</i> .....	53
Figura 13 – Gráfico das métricas de desempenho para o cenário 1 .....	54
Figura 14 - Matriz de confusão da etapa de detecção de falhas do cenário 2 .....	56
Figura 15 - Matriz de confusão do diagnóstico de falhas - Cenários 2 .....	58
Figura 16 - Gráfico do desempenho da etapa de diagnóstico de falhas - Cenário 2 .....	59

## LISTA DE QUADROS

Quadro 1 - Variáveis de processo em um sistema reacional complexo van de Vusse.....	21
Quadro 2 - Parâmetros reacionais do reator CSTR .....	23
Quadro 3 - Cenários de simulação de estado estacionário .....	24
Quadro 4 - Ponto ótimo do sistema de van der Vusse.....	24
Quadro 5 - Perturbações no sistema reacional.....	25
Quadro 6 - Limites das variáveis distúrbio.....	26
Quadro 7 - Limites das variáveis manipuladas.....	26
Quadro 8 - Conjunto de dados com 1 tempo de atraso.....	27
Quadro 9 - Conjunto de dados com 2 tempos de atraso .....	27
Quadro 10 - Conjunto de dados com 3 tempos de atraso .....	27
Quadro 11 - Informações sobre o Benchmark do LABSIA .....	28
Quadro 12 - Representação das variáveis do modelo de RNA.....	34
Quadro 13 - Ferramentas computacionais para criação dos códigos.....	41
Quadro 14 - Falhas aplicadas às variáveis do processo.....	43
Quadro 15 - Métricas de desempenho de classificação .....	48

## LISTA DE TABELAS

Tabela 1 - Tempo para atingir o estado estacionário após as perturbações nas variáveis .....	26
Tabela 2 - Dados de processos após a inserção de falhas.....	44
Tabela 3 - Desempenho dos sensores virtuais em tempos de amostragem e tempos de atraso distintos.....	50
Tabela 4 - Avaliação de métricas por falha do processo do cenário 1 .....	54
Tabela 5 - Avaliação do desempenho do Cenário 1 .....	55
Tabela 6 - Desempenho por processo da etapa de detecção de falhas - Cenário 2 .....	57
Tabela 7 - Desempenho geral da etapa de detecção de falhas - Cenário 2.....	57
Tabela 8 - Avaliação de métricas por falha do processo de diagnóstico de falhas - Cenário 2	58
Tabela 9 - Avaliação geral da etapa de diagnóstico de falhas - Cenário 2 .....	59
Tabela 10 - Comparação do desempenho da etapa de diagnóstico de falhas .....	60

## LISTA DE SIGLAS E ABREVIATURAS

CSTR	<i>Continuous Stirred-Tank Reactor</i>
FDD	<i>Fault Detection and Diagnosis</i>
FN	Falso Negativo
FP	Falso Positivo
IA	Inteligência Artificial
LABSIA	Laboratório de Sistemas Inteligentes Aplicados
LSTM	<i>Long Short-Term Memory</i>
MAE	<i>Mean Absolute Error</i>
MSE	<i>Mean Squared Error</i>
RNA	Rede Neural Artificial
VN	Verdadeiro Negativo
VP	Verdadeiro Positivo

# SUMÁRIO

<b>1</b>	<b>INTRODUÇÃO .....</b>	<b>17</b>
<b>2</b>	<b>OBJETIVOS.....</b>	<b>19</b>
2.1	Objetivo geral .....	19
2.2	Objetivos Específicos .....	19
<b>3</b>	<b>REFERENCIAL TEÓRICO.....</b>	<b>20</b>
3.1	Sistema Reacional Complexo .....	20
3.1.1	Base de dados .....	22
3.1.1.1	Construção do modelo matemático .....	22
3.1.1.2	Determinação do ponto ótimo de operação .....	23
3.1.1.3	Determinação do tempo de amostragem .....	25
3.1.1.4	Dados gerados .....	27
3.2	<i>Deep learning</i> .....	28
3.3	Sensor virtual .....	29
3.4	Detecção e Diagnóstico de Falhas ( <i>FDD</i> ) .....	30
3.5	Redes Neurais Artificiais e Redes <i>Long-Short-Term-Memory (LSTM)</i> .....	32
3.6	Redes Neurais <i>Autoencoder</i> .....	36
3.7	Linguagem Python.....	39
<b>4</b>	<b>METODOLOGIA .....</b>	<b>40</b>
4.1	Implementação em Python.....	41
4.2	Sensor virtual .....	41
4.2.1	Tratamento dos dados.....	42
4.2.2	Criação e treinamento do modelo.....	42
4.2.3	Avaliação do desempenho.....	42
4.3	Detecção e Diagnóstico de Falhas ( <i>FDD</i> ).....	43
4.4	Métricas de desempenho.....	45
4.4.1	Erro Médio Absoluto - <i>MAE</i> .....	45

4.4.2	Erro Quadrático Médio – <i>MSE</i> .....	46
4.4.3	Índice de Jaccard .....	46
4.4.4	Matriz de confusão .....	47
4.5	Apresentação dos resultados .....	48
<b>5</b>	<b>RESULTADOS E DISCUSSÃO .....</b>	<b>50</b>
5.1	Sensor virtual .....	50
5.2	Detecção e Diagnóstico de Falhas .....	52
5.2.1	Cenário 1- Detecção e Diagnóstico de Falhas na mesma etapa .....	53
5.2.2	Cenário 2- Detecção e Diagnóstico de Falhas em etapas separada.....	56
5.2.2.1	Detecção de Falhas .....	56
5.2.2.2	Diagnóstico de falhas.....	57
5.2.3	Comparação dos cenários.....	60
<b>6</b>	<b>CONCLUSÃO .....</b>	<b>61</b>
	<b>REFERÊNCIAS .....</b>	<b>63</b>



# 1 INTRODUÇÃO

A Indústria 4.0 é um termo que descreve o uso de tecnologias digitais avançadas para melhorar a eficiência e a produtividade na indústria. Com o uso de tecnologias como Internet das Coisas (IoT), Big Data e Inteligência Artificial (IA), a Indústria 4.0 permite automação e eficiência elevada nos processos produtivos (SCHWAB, 2016; THIENEN et al., 2016).

Na Indústria de processos químicos, é comum lidar com processos reacionais complexos em que um ou diversos reagentes são submetidos a mais de uma reação química de forma simultânea (CARVALHO, 2017). Um exemplo de processo reacional complexo é o reator de Van de Vusse, que consiste em um modelo matemático para uma reação química em um Reator de Tanque com Agitação Contínua (do inglês *Continuous-flow Stirred Tank Reactors - CSTR*) (VOJTĚŠEK, 2007). Esses processos reacionais são monitorados por sensores físicos tradicionais, que podem sofrer o atraso no recebimento de informações para detectar falhas durante o processo, devido à alta quantidade de monitoramento de dados e suas limitações de processamento.

Nesse contexto, uma das principais tecnologias da Indústria 4.0, que promove automação e eficiência, é o sensor virtual, que permite o monitoramento em tempo real de variáveis como temperatura, pressão, fluxo em processos industriais além da otimização do processo de produção (FORTUNA et al., 2007).

Os sensores virtuais, ao contrário dos sensores físicos, não requerem instalação e manutenção constante, reduzindo os custos com manutenção e aumentando a eficiência do processo produtivo (KADLEC et al., 2009). Ademais, o uso desses sensores em conjunto com técnicas de detecção e diagnóstico de falhas (do inglês, *fault detection and diagnosis – FDD*) é de extrema importância, uma vez que a ocorrência de falhas em processos reacionais complexos pode gerar prejuízos financeiros e riscos à saúde e à segurança (CARVALHO, 2017). Além disso, os sensores virtuais fornecem uma quantidade significativa de dados, que podem ser analisados por meio de técnicas de Inteligência Artificial, em especial, as redes neurais profundas, também conhecidas como *deep learning*, para fornecer informações valiosas sobre o processo de produção (BOSCH, 2022).

A criação de sensores virtuais baseados em técnicas de *deep learning* tem se mostrado uma abordagem promissora para otimização dos processos produtivos industriais (FORTUNA et al., 2007). As técnicas de *deep learning* são capazes de lidar com conjuntos de dados complexos e não lineares, além de identificar padrões ocultos que não seriam detectados por métodos de análise convencionais (GOODFELLOW et al., 2016).

Existem duas principais formas para o aprendizado das técnicas de *deep learning*: aprendizado supervisionado e não supervisionado. O aprendizado supervisionado é uma abordagem de aprendizado de máquina em que um modelo é treinado usando exemplos rotulados. Nesse tipo de aprendizado, o modelo recebe um conjunto de dados de entrada e a saída desejada correspondente a cada exemplo. O objetivo é que o modelo aprenda a mapear corretamente os dados de entrada para suas respectivas saídas (ALLOGHANI). Durante o treinamento, o modelo ajusta seus parâmetros com base nas diferenças entre as saídas previstas e as saídas reais, com o objetivo de minimizar um erro específico. Já o aprendizado não supervisionado é uma abordagem de aprendizado de máquina em que o modelo é treinado em dados não rotulados, ou seja, não há informações explícitas sobre as saídas desejadas. (BRAGA et al., 2000). Nesse tipo de aprendizado, o objetivo é descobrir padrões, estruturas ou relações intrínsecas nos dados de entrada. O modelo busca agrupar os dados de maneira significativa, identificar similaridades ou diferenças entre os exemplos e explorar a estrutura subjacente dos dados.

Nesse contexto, uma técnica de *deep learning* não supervisionada que se destaca é a *Autoencoder*, que consiste em uma rede neural que recebe como entrada um conjunto de dados e gera uma saída com a mesma dimensão da entrada. Essa técnica é utilizada para reconstruir a entrada de forma que ela seja o mais parecida possível com a saída, o que possibilita a identificação de padrões ocultos nos dados (PARK et al., 2019). Essa técnica pode ser aplicada em sensores virtuais para a redução do número de variáveis de entrada, simplificando o processo de análise de dados (GOODFELLOW; BENGIO; COURVILLE, 2016).

Já uma técnica de *deep learning* supervisionada que se destaca é o *Long Short-Term Memory (LSTM)*, que permite a análise de sequências de dados com dependências temporais de longo prazo (LI et al., 2019). Essas técnicas tem sido utilizadas em conjunto, o *Autoencoder* para a detecção de falhas e o *LSTM* para diagnóstico das anomalias. Essas técnicas aplicadas no meio industrial tornam possível a tomada de medidas preventivas e corretivas mais rápidas e eficientes para evitar paradas inesperadas do equipamento e perdas financeiras decorrentes (LI et al., 2019).

Dessa forma, este trabalho buscou utilizar as de técnicas de *deep learning* no desenvolvimento sensores virtuais e na detecção e diagnóstico de falhas para um sistema reacional complexo de van de Vusse, inserindo assim, a aplicabilidade da engenharia química no âmbito da indústria 4.0.

## 2 OBJETIVOS

### 2.1 Objetivo geral

Utilizar técnicas de *deep learning* para desenvolver e validar modelos computacionais, em linguagem Python, de sensores virtuais e estratégias de detecção e diagnóstico de falhas no sistema reacional complexo de van der Vusse em um reator de tanque agitado contínuo (CSTR).

### 2.2 Objetivos Específicos

Para atingir o objetivo geral do presente trabalho, foram escolhidos os seguintes objetivos específicos:

- Aplicar metodologias para desenvolvimento de sensores virtuais utilizando a técnica de *deep learning LSTM*;
- Implementar estratégias de detecção e diagnóstico de falhas utilizando *Autoencoder* e *LSTM*;
- Avaliar o desempenho dos sensores virtuais e dos modelos de detecção e diagnóstico de falhas a partir de métricas de desempenho, comparando os resultados obtidos com o uso das técnicas de *deep learning*.

### 3 REFERENCIAL TEÓRICO

#### 3.1 Sistema Reacional Complexo

Um sistema reacional complexo é um conjunto de elementos que interagem dinamicamente entre si, apresentando comportamentos emergentes que não podem ser previstos a partir das propriedades individuais de seus componentes. Essas interações e retroalimentações são responsáveis pela natureza altamente dinâmica e não-linear desses sistemas (FOGLER, 2009). A complexidade de um sistema reacional é avaliada por meio de diversos fatores, como o número de componentes, a natureza das interações e a dinâmica temporal.

Um exemplo de processo complexo é o reator de Van de Vusse, que consiste em um modelo matemático para uma reação química em um CSTR. O mecanismo dessa reação apresentado por Van de Vusse baseia-se na produção de um composto B, produto de interesse comercial, a partir do consumo do reagente A. São obtidos como subprodutos da reação os compostos C e D, e as respectivas razões de coeficientes ( $K_i$ ) como pode ser observado nas equações 1 e 2 (VAN DE VUSSE, 1964).



As taxas de reação referentes ao consumo do reagente A ( $r_A$ ) e à formação do produto B ( $r_B$ ), são descritas pelas Equações 3 e 4.

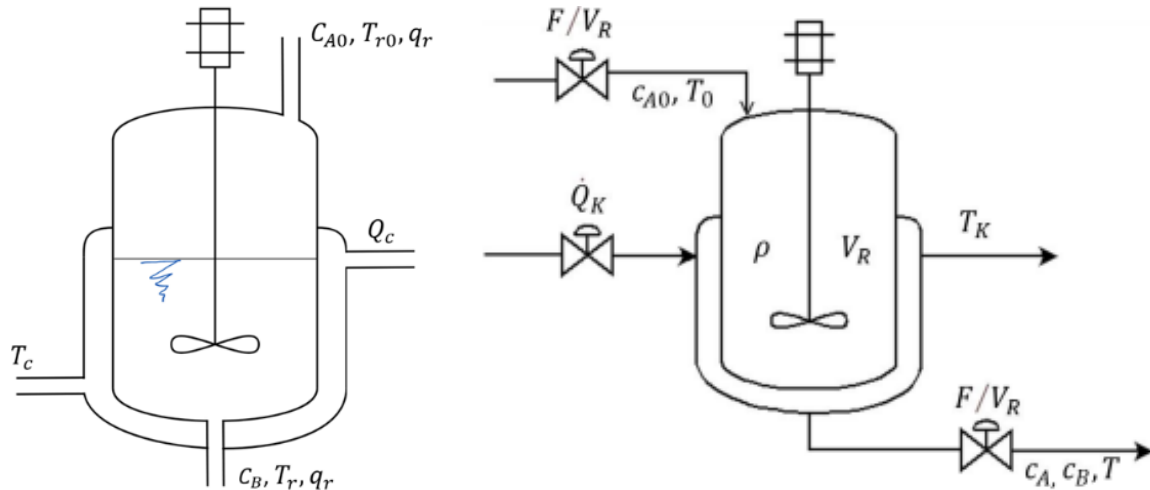
$$r_A = k_1 C_A + k_3 C_A^2 \quad (3)$$

$$r_B = k_1 C_A + k_2 C_B \quad (4)$$

O sistema é alimentado por uma corrente com vazão volumétrica  $q_r$ , concentração de reagente  $C_{A0}$  e temperatura  $T_{r0}$ . O controle da temperatura do reator é feito por uma jaqueta térmica (KLATT & ENGELL, 1998). Para controlar a vazão volumétrica e a temperatura da corrente de alimentação da jaqueta térmica, é utilizado um trocador de calor externo. Esse trocador de calor é responsável pela troca térmica entre o fluido refrigerante do trocador e o fluido refrigerante que alimenta o sistema do reator. A vazão volumétrica e a temperatura da corrente de alimentação do fluido refrigerante da jaqueta térmica são controladas por  $Q_c$ .

O sistema funciona em regime contínuo e apresenta uma corrente de saída do reator com temperatura  $T_r$  e vazão volumétrica  $q_r$ . A corrente de saída da jaqueta térmica possui temperatura  $T_c$ . A representação do esquema de um reator de van de Vusse é ilustrada na figura 1.

**Figura 1-** Representação esquemática de um reator de van de Vusse



Fonte: Klatt & Engell (1998)

O quadro 1 apresenta as variáveis presentes no processo reacional complexo de van der Vusse.

**Quadro 1 -** Variáveis de processo em um sistema reacional complexo van de Vusse

Variável	Representação	Unidade de Medida
$C_{A0}$	Concentração de reagente A na alimentação	$kmol/m^3$
$C_A$	Concentração de A no meio reacional	$kmol/m^3$
$C_B$	Concentração de B no meio reacional	$kmol/m^3$
$Q_c$	Troca térmica no trocador de calor	$kJ/min$
$q_r$	Vazão volumétrica da corrente de alimentação	$m^3/min$
$T_c$	Temperatura da corrente de saída da jaqueta térmica	$K$
$T_r$	Temperatura do meio reacional	$K$
$T_{r0}$	Temperatura da corrente de alimentação	$K$

Fonte: Vojtěšek (2007)

O balanço de massa e de energia por componente para o reator CSTR no estado estacionário resulta nas equações:

$$\frac{dC_A}{dt} = \frac{q_r}{V_r} (C_{A0} - C_A) - K_1 C_A - K_3 C_A^2 \quad (5)$$

$$\frac{dC_B}{dt} = \frac{q_r}{V_r} C_B + K_1 C_A - K_2 C_B \quad (6)$$

$$\frac{dT_c}{dt} = \frac{1}{m_c C_{Pc}} [Q_c + A_r U (T_c - T_r)] \quad (7)$$

$$\frac{dT_r}{dt} = \frac{q_r}{V_r} (T_{r0} - T_r) + \frac{h_R}{C_{pr} \rho_r} + \frac{A_r U}{V_r \rho_r C_{pr}} (T_c - T_r) \quad (8)$$

onde  $C_A \geq 0$ ,  $C_B \geq 0$ .

É possível notar a não linearidade do sistema, que pode ser encontrada nas taxas de reação ( $k_j$ ), que são descritas pela lei de Arrhenius na equação 9:

$$k_j = k_{0j} \cdot \exp\left(\frac{E_j}{RT_j}\right), \text{ para } j = 1, 2, 3 \dots \quad (9)$$

Onde,  $k_{0j}$  representam os fatores pré-exponenciais e  $E_j$  são as energias de ativação.

O calor de reação ( $h_R$ ) na equação 10 é expresso como

$$h_r = h_1 \cdot k_1 \cdot c_A + h_2 \cdot k_2 \cdot c_B + h_3 \cdot k_3 \cdot c_A^2 \quad (10)$$

Onde  $h_i$  representam as entalpias de reação.

### 3.1.1 Base de dados

A base de dados para a construção dos sensores virtuais e das estratégias de detecção e diagnóstico de falhas, baseada na modelagem matemática do processo reacional complexo de Van de Vusse em um reator CSTR proposto por Vojtesek (2007), é um benchmark desenvolvido pelo Laboratório de Sistemas Inteligentes Aplicados (LABSIA). A base de dados é composta por mais de 24.000.000 de dados do processo, com 8 variáveis monitoradas, tornando o problema dentro do contexto de big data.

#### 3.1.1.1 Construção do modelo matemático

A construção do modelo matemático do processo reacional complexo de van der Vusse em um reator CSTR utilizou os parâmetros reacionais e de projeto do reator CSTR do trabalho desenvolvido por Vojtesek (2007), conforme apresenta o quadro 2.

**Quadro 2** - Parâmetros reacionais do reator CSTR

Nome do parâmetro	Símbolo e valor do parâmetro
Volume do reator	$V_r=0,01 \text{ m}^3$
Densidade do reagente	$r=934,2 \text{ kg.m}^3$
Capacidade calorífica do reagente	$c_{pr}=3,01 \text{ kJ.kg}^{-1}.\text{K}^{-1}$
Massa do fluido refrigerante	$m_c=5 \text{ kg}$
Capacidade calorífica do fluido refrigerante	$c_{pc}=2,0 \text{ kJ.kg}^{-1}.\text{K}^{-1}$
Superfície de troca térmica	$A_r=0,215 \text{ m}^2$
Coeficiente de transferência de calor	$U=67,2 \text{ kJ.min}^{-1}.\text{m}^{-2}.\text{K}^{-1}$
Fator pré-exponencial para a reação 1	$k_{01}=2,145 \cdot 10^{10} \text{ min}^{-1}$
Fator pré-exponencial para a reação 2	$k_{02}=2,145 \cdot 10^{10} \text{ min}^{-1}$
Fator pré-exponencial para a reação 3	$k_{03}=1,5072 \cdot 10^8 \text{ min}^{-1} \text{ kmol}^{-1}$
Razão entre a energia de ativação e R para a reação 1	$E_{1R}=9758,3 \text{ K}$
Razão entre a energia de ativação e R para a reação 2	$E_{2R}=9758,3 \text{ K}$
Razão entre a energia de ativação e R para a reação 3	$E_{3R}=8560 \text{ K}$
Entalpia da reação 1	$h_1= -4200 \text{ kJ.kmol}^{-1}$
Entalpia da reação 2	$h_2=11000 \text{ kJ.kmol}^{-1}$
Entalpia da reação 3	$h_3=41850 \text{ kJ.kmol}^{-1}$
Concentração de A na corrente de alimentação	$c_{A0}=5,1 \text{ kmol.m}^{-3}$
Temperatura da corrente de alimentação de reagentes	$T_{r0}=387,05 \text{ K}$

Fonte: Vojtěšek (2007)

### 3.1.1.2 Determinação do ponto ótimo de operação

O estado estacionário é alcançado em sistemas estáveis quando as variáveis de entrada permanecem constantes ao longo do tempo, e esse estado é atingido à medida que o tempo tende ao infinito. No estado estacionário, as variações nas variáveis são igualadas a zero. O estado estacionário é caracterizado exclusivamente pelos valores das variáveis de vazão volumétrica da corrente de alimentação ( $q_r$ ), concentração do reagente A na corrente de alimentação ( $c_{A0}$ ), temperatura da corrente de alimentação ( $T_{r0}$ ) e a quantidade de energia térmica removida do fluido refrigerante antes de ser alimentado na jaqueta térmica ( $Q_c$ ).

As condições iniciais, incluindo a concentração inicial do reagente A ( $C_a$  inicial), concentração inicial do produto B ( $C_b$  inicial), temperatura inicial do reator ( $T_r$  inicial) e

temperatura inicial do fluido refrigerante ( $T_c$  inicial), não têm impacto nos valores das variáveis no estado estacionário.

Considerando que os valores das variáveis de processo ( $c_A$ ,  $c_B$ ,  $T_r$ ,  $T_c$ ) dependem apenas das variáveis de entrada  $c_{A0}$ ,  $T_{r0}$ ,  $q_r$  e  $Q_c$ , foram realizadas simulações para diferentes cenários de  $Q_c$  e  $q_r$ , com o objetivo de analisar o estado estacionário e definir o ponto ótimo de operação do reator. Os diferentes cenários estão apresentados no quadro 3.

**Quadro 3** - Cenários de simulação de estado estacionário

Variável	Cenários de simulação de estado estacionário
$Q_c$	$\langle -500;500 \rangle$ kJ.min <sup>-1</sup>
$q_r$	$\langle 0,001;0,030 \rangle$ m <sup>3</sup> .min <sup>-1</sup>
$c_{A0}$	5,1 kmol.m <sup>3</sup>
$T_{r0}$	387,05 K

Fonte: LABSIA (2023)

A seleção do ponto ótimo de operação foi determinada pela concentração do produto B,  $C_B$ , e visou maximizar a produção desse composto no sistema. A otimização foi realizada com algoritmos genéticos, atingindo o ponto ótimo de operação apresentado no quadro 4.

**Quadro 4** - Ponto ótimo do sistema de van der Vusse

Variável	Valor de E.E.
$Q_c$	144,5 kJ/min
$q_r$	0,0032 m <sup>3</sup> /min
$C_{A0}$	5,1 kmol/m <sup>3</sup>
$T_{r0}$	387,05 K
$C_A^S$	2,1944 kmol/m <sup>3</sup>
$C_B^S$	1,0950 kmol/m <sup>3</sup>
$T_r^S$	391,38 K
$T_c^S$	401,39 K

Fonte: LABSIA (2023)



### 3.1.1.3 Determinação do tempo de amostragem

O tempo de amostragem desempenha um papel crucial na qualidade da representação do perfil dinâmico. Nesse sentido, são adotadas as diretrizes propostas por Meleiro (2002). Segundo o autor, é importante evitar tempos de amostragem excessivamente reduzidos, uma vez que podem resultar em problemas relacionados ao condicionamento numérico. Por outro lado, tempos de amostragem muito longos acarretam perda de informação dinâmica do processo, o que compromete a representatividade dos dados. Portanto, para determinar o tempo de amostragem adequado, recomenda-se a realização de estudos sobre a dinâmica do processo, considerando diferentes entradas e perturbações.

Adicionalmente, foi levada em consideração a recomendação de Cooper (2007) citada por Silva (2014), que estabelece que o tempo de amostragem não deve ultrapassar 10% da constante de tempo do sistema, como uma prática recomendada. Seguindo a abordagem adotada por Silva (2014), foram definidos três tempos de amostragem com o intuito de avaliar seu impacto na predição de técnicas de aprendizado de máquina.

Para a definição dos tempos de amostragem, em cada variável avaliada, foi registrado o tempo que o sistema leva para atingir um novo estado estacionário após cada perturbação e o comportamento de  $C_B$ ,  $C_A$ ,  $T_r$  e  $T_c$  após a perturbação. O objetivo desta análise é determinar o melhor tempo de amostragem, de modo a garantir que não haja perdas de informações sobre a dinâmica do processo. Os limites superiores e inferiores das perturbações para cada variável são apresentados no quadro 5.

**Quadro 5** - Perturbações no sistema reacional

<b>Variável</b>	<b>Perturbação superior</b>	<b>Perturbação inferior</b>
$C_{A0}$	+10%	-10%
$Q_c$	+10%	-10%
$q_r$	+10%	-10%
$T_{r0}$	+2,58%	-2,58%

**Fonte:** LABSIA (2023)

No que diz respeito às variáveis de entrada, foram utilizados os limites estabelecidos nos quadros 6 e 7.

**Quadro 6** - Limites das variáveis distúrbio

<b>Variáveis Distúrbio</b>		
<b>Limites</b>	CA0:(4,59:5,61)	Tr0: 377,05:397,05
<b>Unidade</b>	kmol/m <sup>3</sup>	K

Fonte: LABSIA (2023)

**Quadro 7** - Limites das variáveis manipuladas

<b>Variáveis Manipuladas</b>		
<b>Limites</b>	Qc:(130,05:158,95)	qr: 0,00288:0,00352
<b>Unidade</b>	kJ/min	m <sup>3</sup> /min

Fonte: LABSIA (2023)

Após as perturbações, foi analisado o tempo necessário para o sistema atingir o estado estacionário, a partir do comportamento da concentração de B. A tabela 1 apresenta os valores das variáveis com as perturbações e os respectivos tempos para o novo estado estacionário.

**Tabela 1** - Tempo para atingir o estado estacionário após as perturbações nas variáveis

<b>Variável</b>	<b>Perturbação superior</b>	<b>Tempo para o novo E.E</b>
q <sub>r</sub> (+10%)	0,00352 m <sup>3</sup> /min	491,64 s
q <sub>r</sub> (+10%)	0,00288 m <sup>3</sup> /min	620,81 s
Q <sub>c</sub> (+10%)	158,95 kJ/min	579,30 s
Q <sub>c</sub> (+10%)	130,05 kJ/min	571,80 s
C <sub>A0</sub> (+10%)	5,61 kmol/m <sup>3</sup>	700,20 s
C <sub>A0</sub> (+10%)	4,59 kmol/m <sup>3</sup>	687,60 s
T <sub>r0</sub> (+10%)	397,05 K	1126,00 s
T <sub>r0</sub> (+10%)	377,05 K	1031,40 s

Fonte: LABSIA (2023)

Após a análise do comportamento de CB, notou-se o pico de produção em 65 segundos e levando em consideração esse tempo, foram definidos 4 tempos de amostragem para estudo da dinâmica do sistema simulado, sendo eles: 3 segundos, 6 segundos, 9 segundos e 12 segundos.

### 3.1.1.4 Dados gerados

Após a simulação computacional, foram gerados quatro conjuntos de dados, com tempos de amostragem diferentes para cada conjunto. Os quadros 8, 9 e 10 apresentam a organização dos conjuntos de dados e seus respectivos tempos de atraso.

**Quadro 8** - Conjunto de dados com 1 tempo de atraso

Conjunto	Entrada – 1 Atraso	Saída
C1-1 (3 segundos)	CA0(k-1) CB(k-1) Qc(k-1) qr(k-1) Tr(k-1) TC(k-1) Tr0(k-1)	CB(k)
C1-2 (6 segundos)		
C1-3 (9 segundos)		
C1-4 (12 segundos)		

Fonte: LABSIA (2023)

**Quadro 9** - Conjunto de dados com 2 tempos de atraso

Conjunto	Entrada – 2 Atraso		Saída
C2-1 (3 segundos)	CB(k-1) Tr(k-1)	CB(k-2) Tr(k-2)	CB(k)
C2-2 (6 segundos)	TC(k-1) Tr0(k-1)	TC(k-2) Tr0(k-2)	
C2-3 (9 segundos)	CA0(k-1) Qc(k-1)	CA0(k-2) Qc(k-2)	
C2-4 (12 segundos)	qr(k-1)	qr(k-2)	

Fonte: LABSIA (2023)

**Quadro 10** - Conjunto de dados com 3 tempos de atraso

Conjunto	Entrada – 3 Atrasos			Saída
C3-1 (3 segundos)	CB(k-1) Tr(k-1)	CB(k-2) Tr(k-2)	CB(k-3) Tr(k-3)	CB(k)
C3-2 (6 segundos)	TC(k-1) Tr0(k-1)	TC(k-2) Tr0(k-2)	TC(k-3) Tr0(k-3)	
C3-3 (9 segundos)	CA0(k-1) Qc(k-1)	CA0(k-2) Qc(k-2)	CA0(k-3) Qc(k-3)	
C3-4 (12 segundos)	qr(k-1)	qr(k-2)	qr(k-3)	

Fonte: LABSIA (2023)

Os dados de processos proveniente da simulação matemáticas estão armazenados em um banco de dados relacional do LABSIA. O quadro 11 apresenta a quantidade de dados de processos.

**Quadro 11** - Informações sobre o Benchmark do LABSIA

<b>Descrição</b>	<b>Quantidade</b>	<b>Tamanho</b>
Pertubações em $T_{r0}$	24.995	-
Pertubações em $C_{A0}$	24.988	-
Pertubações em $Q_c$	24.788	-
Pertubações em $q_r$	25.020	-
Base de 3 segundos	11.962.180	1,594 GB
Base de 6 segundos	5.981.090	0,717 GB
Base de 9 segundos	3.987.393	0,477 GB
Base de 12 segundos	2.990.545	0,358 GB

**Fonte:** LABSIA (2023)

### **3.2 Deep learning**

A *deep learning* é uma subárea da *manchine learning* que tem como objetivo criar modelos capazes de aprender a partir de dados complexos e de alta dimensionalidade. Esses modelos são compostos por redes neurais artificiais profundas, capazes de aprender a representar os dados de forma hierárquica, extraindo características cada vez mais abstratas em camadas mais profundas (BOCHIE et al., 2020).

As redes neurais artificiais profundas são compostas por diversas camadas de neurônios, que são unidades de processamento que recebem entradas, realizam operações matemáticas e geram saídas. Cada camada da rede realiza uma transformação não-linear nos dados de entrada, permitindo que a rede aprenda a representar características cada vez mais abstratas (GOODFELLOW; BENGIO; COURVILLE, 2016). A camada de entrada recebe os dados brutos, como imagens ou áudio, e as camadas subsequentes processam esses dados, transformando-os em representações mais complexas.

As técnicas de *deep learning* se dividem basicamente em duas principais abordagens: o aprendizado supervisionado e não supervisionado.

- **Aprendizado supervisionado:** No aprendizado supervisionado, o modelo é treinado com um conjunto de dados rotulados, ou seja, dados que já possuem uma classificação ou resultado esperado. Isso permite que o modelo aprenda a fazer previsões precisas com base em novos dados de entrada (BOCHIE et al., 2020). Esse tipo de abordagem essa abordagem é comumente empregada em situações onde há uma grande quantidade de dados disponíveis e os valores de saída do processo são conhecidos.
- **Aprendizado não supervisionado:** É utilizado quando não há um conjunto de dados rotulados disponíveis. Nessas situações, o objetivo é identificar padrões ou estruturas ocultas nos dados, o que pode ser útil para identificar problemas em processos produtivos ou para a detecção de anomalias (GOODFELLOW; BENGIO; COURVILLE, 2016). Essa abordagem é mais adequada para problemas em que não há uma saída nítida ou quando o conjunto de dados é muito grande para ser rotulado manualmente.

Apesar de suas muitas aplicações, o *deep learning* ainda é uma área de pesquisa em constante evolução, com muitos desafios a serem superados. Entre eles, destacam-se o problema da generalização, ou seja, a capacidade da rede neural de se adaptar a novos dados que não foram vistos durante o treinamento, e o problema da interpretabilidade, ou seja, a capacidade de entender como a rede neural tomou suas decisões (MEHDIYEV et al., 2017)

### 3.3 Sensor virtual

O termo *softsensor* ou sensor virtual, refere-se a modelos matemáticos capazes de estimar valores de variáveis de processo a partir de outras variáveis disponíveis em um sistema de monitoramento (KADLEC; GABRYS; STRANDT, 2009b). A técnica de Sensor virtual é amplamente utilizada na indústria para obter informações em tempo real sobre processos e produtos, permitindo melhorias significativas em eficiência e qualidade.

O surgimento dos sensores virtuais no âmbito industrial, surgiu da complexidade dos processos e da dificuldade em obter informações precisas em tempo real na indústria, além disso, tornou-se cada vez mais importante a capacidade de detectar e corrigir falhas rapidamente, a fim de minimizar o impacto na produção e na qualidade do produto final. Sensores convencionais podem não ser suficientes para monitorar variáveis críticas, além de

serem caros e requererem manutenção constante (SHAO; TIAN, 2017). Sensores virtuais, por outro lado, podem ser desenvolvidos a partir de dados existentes e usados para prever valores de variáveis não medidas, permitindo que os operadores obtenham informações em tempo real e tomem decisões mais informadas (WANG; LI; HAN, 2023).

Os benefícios da implementação de sensores virtuais incluem a melhoria da eficiência do processo, redução de custos, aumento da qualidade do produto e melhorias na segurança do processo. Além disso, o uso de Sensor virtual pode ajudar a reduzir o tempo de desenvolvimento e implantação de novos processos, uma vez que a necessidade de sensores físicos adicionais é reduzida (GONZAGA et al., 2009).

Um campo importante para a aplicação dos sensores virtuais são os de monitoramento e detecção de falhas de processos, em que os sensores desempenham a função de detectar o estado do processo e na ocorrência de desvio das condições normais identificam a causa do desvio (KADLEC; GABRYS; STRANDT, 2009b). A detecção e diagnóstico de falhas com sensores virtuais é baseada em modelos matemáticos empíricos que utilizam dados históricos de variáveis de processo para prever o comportamento futuro. Esses modelos são treinados utilizando técnicas de *deep learning*, e podem ser usados para prever o comportamento de variáveis não medidas, identificar possíveis falhas e sugerir medidas corretivas (KADLEC; GABRYS; STRANDT, 2009a).

Devido as vantagens de baixo custo e ao atraso de medições insignificantes, diversos algoritmos de modelagem têm sido aplicados no desenvolvimento de sensores virtuais em muitas indústrias de processo, como o de coluna de destilação, o processo de tratamento de efluentes, dentre outros (SHAO; TIAN, 2017).

No entanto, a implementação de sensores virtuais também apresenta desafios técnicos. Um dos principais desafios é o desenvolvimento de modelos matemáticos precisos capazes de estimar com precisão as variáveis de interesse. Os modelos de sensores virtuais também precisam ser atualizados regularmente para garantir que reflitam as mudanças no processo (WANG et al., 2019). A qualidade dos dados também pode ser um desafio, pois os modelos de sensores virtuais dependem de dados precisos e confiáveis.

### **3.4 Detecção e Diagnóstico de Falhas (FDD)**

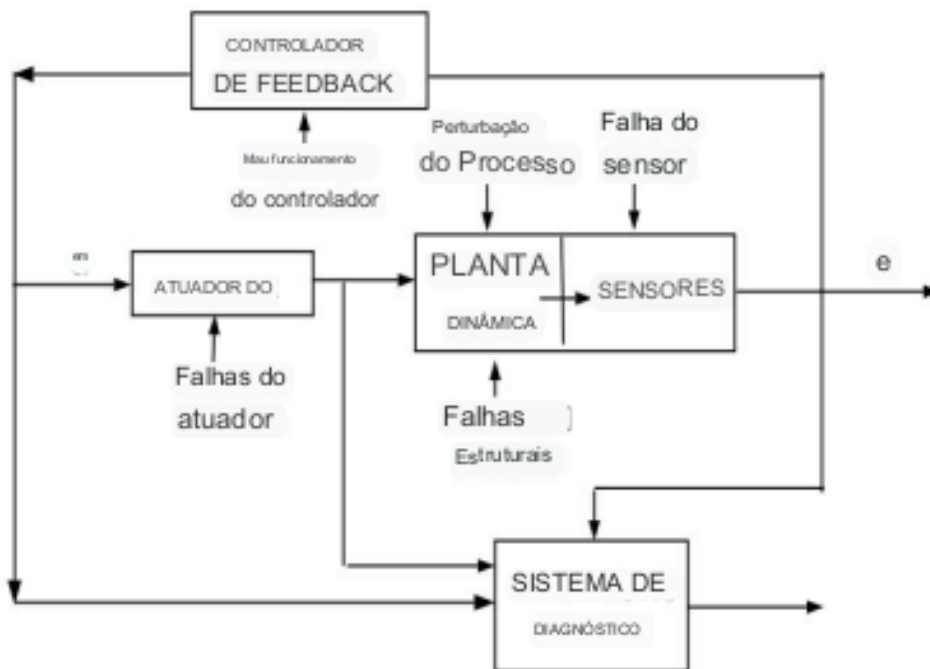
A Detecção e diagnóstico de falhas (FDD - *Fault Detection and Diagnosis*) é um processo que envolve a identificação e análise de sinais, dados e informações de um sistema,

com o objetivo de detectar a presença de falhas e determinar a causa subjacente. O termo falha está atrelado ao distanciamento de um intervalo aceitável de um parâmetro associado a um processo (VENKATASUBRAMANIAN et al., 2003).

Em geral, o processo é dividido em duas etapas: detecção de falhas, que envolve a identificação da presença de uma anomalia no sistema, e diagnóstico de falhas, que envolve a identificação da causa raiz da falha (RUSSELL et al., 2000). A detecção de falhas é uma etapa fundamental no processo de manutenção preventiva de sistemas industriais e mecânicos. Essa etapa envolve a identificação da presença de uma anomalia no sistema, que pode ser indicada por meio de dados de sensores, alarmes ou outros sinais de alerta (PARK et al., 2019). Esses dados são processados por algoritmos de aprendizado de máquina, que podem ser treinados para identificar padrões anormais para indicar a presença de uma falha iminente.

Após a detecção de uma falha, a próxima etapa é o diagnóstico, que envolve a identificação da causa raiz da anormalidade. Nessa etapa, torna-se necessário analisar detalhadamente os dados coletados na etapa de detecção, a fim de determinar a causa subjacente da anomalia (DAI & GAO, 2013). O diagnóstico pode envolver a análise de dados históricos, inspeção visual da máquina e outras técnicas de análise (GAO; CECATI; DING, 2015). A figura 2, apresenta um sistema de processo controlado e indica as diferentes fontes de falhas nele.

**Figura 2** - Fontes de falhas em um processo



**Fonte:** Venkatasubramanian et al. (2003)

Em geral, é preciso lidar com três classes de falhas: Mudanças brutas de parâmetros em um modelo, que se refere as falhas de parâmetros que surgem quando ocorre uma perturbação que entra no processo vinda do ambiente por meio de uma ou mais variáveis independentes (GAO; CECATI; DING, 2015); mudanças estruturais, que se refere as alterações dentro do próprio processo, através de falhas em equipamentos, gerando mudanças no fluxo de informações entre várias variáveis. Sensores e atuadores com defeitos, referem-se as falhas dos instrumentos que fornecem sinais de *feedback*, que são essenciais para o controle da planta. Essas falhas ocasionam desvio dos limites aceitáveis das variáveis do processo (VENKATASUBRAMANIAN et al., 2003)

As abordagens para se desenvolver estratégias de *FDD* podem se dividir em dois principais grupos: métodos usando dados históricos de processo e métodos utilizando modelos determinísticos (SOARES, 2017). A abordagem baseada em dados *data-driven*, por proporcionar diagnósticos satisfatórios e não necessitarem de descrições complexas dos modelos fenomenológicos, tem atraído pesquisadores e indústrias nos últimos anos (SAUFI et al., 2019).

As estratégias mais tradicionais de detecção e diagnóstico de falhas baseadas em dados usam métodos de controle estatístico de processo multivariado (MSPC), contudo esses métodos apresentam limitações devido a essas técnicas serem lineares, resultando em desempenho abaixo do adequado quando se trata de processos complexos (PARK et al., 2019). Diante dessas limitações, as redes neurais são a alternativa para a contornar os problemas das abordagens tradicionais na realização dos diagnósticos de falhas.

As técnicas *deep learning*, e suas variantes foram desenvolvidas para lidar com problemas sequenciais complexos. Em particular, a rede *LSTM* extrai efetivamente as dependências temporais de longo prazo, junto com as de curto prazo para dados de séries temporais usando portas não lineares. Já o *autoencoder*, extrai as características do estado normal para realizar a detecção de eventos raros. Além disso, o *autoencoder* após o aprendizado, detecta quaisquer novos dados que venham do estado normal do processo, pois terão o mesmo padrão ou distribuição (PARK et al., 2019).

### **3.5 Redes Neurais Artificiais e Redes *Long-Short-Term-Memory* (*LSTM*)**

As redes neurais artificiais (RNAs) são uma classe de algoritmos de aprendizado de máquina inspirados no funcionamento do sistema nervoso central humano. Elas são capazes de



aprender a partir de exemplos e reconhecer padrões complexos em dados (SOARES; DA SILVA, 2011). As redes neurais são consideradas uma das principais técnicas de aprendizado de máquina, sendo amplamente empregadas em áreas como reconhecimento de imagem, processamento de fala, previsão de séries temporais e muitas outras aplicações.

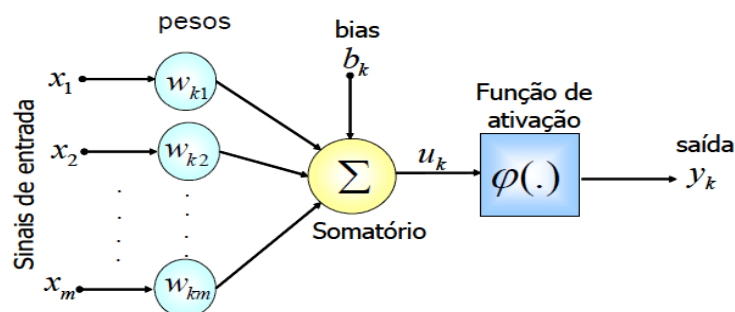
As redes neurais são compostas por um conjunto de neurônios artificiais organizados em camadas, capazes de processar um conjunto de entradas e gerar uma ou mais saídas. Cada neurônio é conectado a outros neurônios em camadas subsequentes, formando uma rede complexa de interconexões (HAYKIN, 2009). Durante o processo de treinamento da rede, os pesos das conexões são ajustados de forma iterativa para que a rede possa reconhecer padrões nos dados de entrada.

O treinamento de uma rede neural requer a definição de uma função de perda que avalie a diferença entre a saída da rede e o resultado desejado. O objetivo do treinamento é minimizar essa função de perda, ajustando os pesos das conexões para melhorar a precisão da rede (KINGMA & BA, 2014).

Embora as redes neurais sejam uma técnica poderosa de aprendizado de máquina, é importante destacar que elas requerem grande quantidade de dados e tempo de treinamento. Além disso, a escolha da arquitetura da rede e da inicialização dos pesos das conexões pode ter um impacto significativo na precisão da rede.

As Redes neurais distinguem-se pela arquitetura e como os pesos sinápticos associados às conexões são adaptados durante o processo de aprendizado. A arquitetura de uma RNA limita os tipos de problemas que a rede pode usar e é definida pelo número de camadas (camadas simples ou múltiplas), o número de nós em cada camada, o tipo de conexões entre os nós (*feedforward* ou *feedback*), e sua topologia (SOARES; DA SILVA, 2011). A figura 3 apresenta a estrutura de um neurônio artificial, também conhecido como *perceptron*, já o quadro 12 é a apresentação do significado das variáveis da RNA.

**Figura 3 - Modelo de uma RNA**



**Fonte:** Soares & da Silva (2011)

**Quadro 12** - Representação das variáveis do modelo de RNA

<b>Representação das variáveis da RNA</b>	
$bk$	Bias
$uk$	Combinação linear dos sinais de entrada
$wkm$	Pesos sinápticos
$xm$	Entradas da rede
$yk$	Saída do neurônio
$\varphi(\cdot)$	Função de ativação

**Fonte:** Soares & da Silva (2011)

Os pesos sinápticos são os parâmetros que se adaptam de acordo com apresentação do conjunto de treinamento à rede. O termo “bias”, representado por  $bk$ , é o efeito direto na função de ativação na entrada da rede, tendo o efeito de acréscimo ou decréscimo da função. Além disso, o “bias” tem a utilidade de melhorar a adaptação do neurônio, aumentando o grau de liberdade (Soares & da Silva, 2011).

Um neurônio artificial consiste em receber as entradas  $xm$ , realizar somas ponderadas com os pesos sinápticos  $wkm$ , adicionar um viés, representado pela constante  $bk$ , e aplicar a função de ativação  $\varphi(\cdot)$  com a finalidade de obter a saída  $yk$ . A equação 11, 12 e 13 representam matematicamente o processo.

$$uk = \sum_{k=1}^n wk \cdot xk + b \quad (11)$$

$$vk = uk + bk \quad (12)$$

$$yk = \varphi(vk) \quad (13)$$

A função de ativação define a saída do neurônio em termos do campo local induzindo  $vk$  (HAYKIN, 2001). As funções de ativações que comumente são usadas e as que mais se destacam estão expressas nas equações 14, 15, 16 e 17.

$$ReLU(x) = \max(0, x) \quad (14)$$

$$Sigmoid(x) = \frac{1}{1+e^{-x}} \quad (15)$$

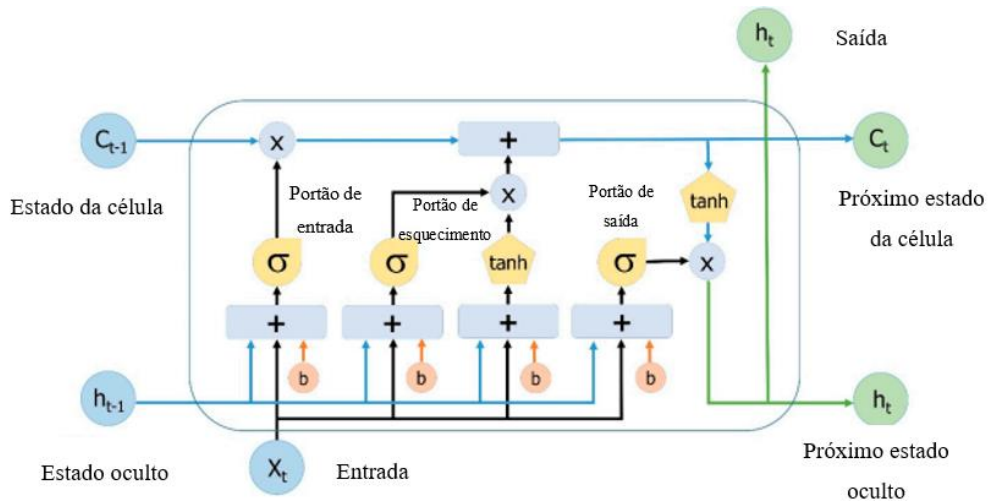
$$Softmax(x) = \frac{\exp(xi)}{\sum_k \exp(xi)} \quad (16)$$

$$Tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (17)$$

Além das redes neurais tradicionais, destaca-se as redes neurais *LSTM*, uma classe de redes neurais recorrentes que foram projetadas para lidar com problemas de seqüências de dados longas, com ênfase na previsão e modelagem de séries temporais (LI et al., 2019).

A estrutura das redes neurais *LSTM*, conforme apresenta a figura 4, consiste em células de memória, cada uma delas contendo três portas principais para controlar o fluxo de informações: entrada (*input*), esquecimento (*forget*) e saída (*output*).

**Figura 4** - Estrutura de RNA *LSTM*



Fonte:(LE et al., 2019)

- **Portão de entrada (*input gate*):** O portão de entrada é responsável por selecionar as informações relevantes que devem ser adicionadas à célula de memória. Isso é feito por meio da multiplicação ponto a ponto da saída da camada anterior com os pesos da porta de entrada, seguido de uma ativação sigmoide que decide quais informações passarão para a célula de memória (NI et al., 2020). A porta de entrada tem a capacidade de detectar características importantes de entrada e de adicionar novas informações relevantes para o estado atual da célula de memória. Matematicamente, conforme apresentam as equações 18 e 19, a informação é regulada usando a função sigmoide, filtrando os valores a serem lembrados usando as entradas  $x_t$  e  $h_{t-1}$ .

$$i_t = \sigma(w_3 x_t + w_4 h_{t-1} + b) \quad (18)$$

$$\tilde{c}_t = \tanh(w_5 x_t + w_6 h_{t-1} + b) \quad (19)$$

- **Portão de esquecimento (*forget gate*):** O portão de esquecimento é responsável por decidir quais informações devem ser descartadas da célula de memória. Isso é feito por

meio da multiplicação ponto a ponto da saída da camada anterior com os pesos da porta de esquecimento, seguido de uma ativação sigmoide que decide quais informações serão esquecidas, conforme apresenta a equação 20. A porta de esquecimento tem a capacidade de aprender quais informações são menos relevantes para o estado atual da célula de memória e descartá-las, o que ajuda a manter a célula de memória limpa e relevante (LV et al., 2022).

$$f_t = \sigma(W_1 x_t + W_2 h_{t-1} + b) \quad (20)$$

- **Portão de saída (*output gate*):** O portão de saída é responsável por decidir qual é a saída da célula de memória, ou seja, quais informações serão enviadas para a próxima camada da rede ou para a saída final da rede. Isso é feito por meio da multiplicação ponto a ponto da saída da camada anterior com os pesos da porta de saída, seguido de uma ativação sigmoide que decide quais informações serão enviadas para a próxima camada da rede. A porta de saída tem a capacidade de selecionar as informações mais relevantes da célula de memória e usá-las para gerar uma saída precisa e relevante, conforme apresentado nas equações 21, 22 e 23. O operador  $\odot$  representa o produto ponto a ponto entre os vetores, denominado produto de Hadamard (PARK et al, 2019).

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \quad (21)$$

$$o_t = \sigma(W_7 x_t + W_8 h_{t-1} + b) \quad (22)$$

$$h_t = o_t \odot \tanh(c_t) \quad (23)$$

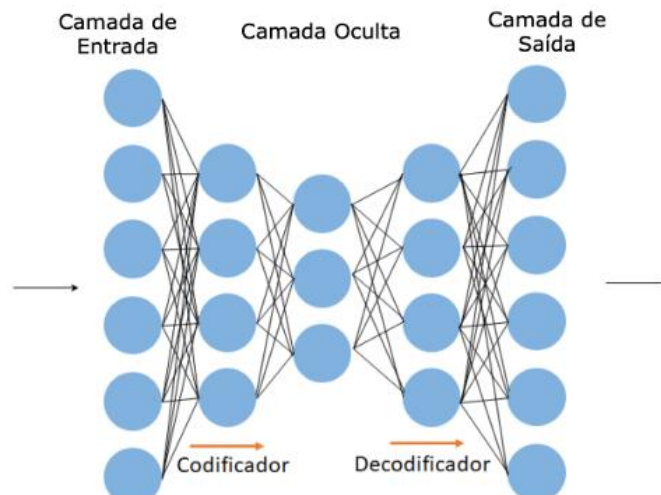
Devido a capacidade do *LSTM* em manipular sequências de dados e identificar dependências de longo prazo torna-o extremamente adequado para modelar o comportamento de sistemas dinâmicos. Dessa forma, o *LSTM* pode ser aplicado na identificação de padrões sutis presentes nos dados de sensores, o que torna eficiente nas estratégias de *FDD* (ZHAO et al., 2018).

### 3.6 Redes Neurais *Autoencoder*

O *autoencoder* é uma técnica de *deep learning* não supervisionado que é utilizada para reduzir a dimensionalidade dos dados, extrair características importantes e reconstruir a entrada original a partir dessas características. Durante a extração de recursos, o *autoencoder* pode reter

a maior parte das informações de entrada, minimizando o erro de reconstrução (Luo *et al.*, 2019). Essa técnica é composta por duas partes principais: o *encoder* e o *decoder*. A estrutura geral de uma Rede Neural *Autoencoder* é representada na figura 5.

**Figura 5** - Estrutura de uma rede neural Autoencoder



**Fonte:** Esteves (2020)

O *encoder* é a primeira parte da rede neural e tem como objetivo reduzir a dimensionalidade dos dados de entrada. Ele consiste em várias camadas que transformam a entrada original em um espaço latente de menor dimensão. Cada camada do *encoder* extrai características relevantes da entrada original e reduz sua dimensionalidade (LEE *et al.*, 2021). As camadas do *encoder* geralmente têm funções de ativação não lineares, como ReLU (*Rectified Linear Unit*), para permitir que a rede neural aprenda relações mais complexas nos dados de entrada (PARK *et al.*, 2019).

O *decoder* é a segunda parte da rede neural e tem como objetivo reconstruir a entrada original a partir das características extraídas pelo *encoder*. O *decoder* é composto por várias camadas que transformam a representação latente de volta à dimensão original. Cada camada do *decoder* expande a dimensionalidade das características extraídas pelo *encoder* e reconstrói uma versão aproximada da entrada original (VINCENT *et al.*, 2010). O *decoder* também usa funções de ativação não lineares, como ReLU ou sigmoid, para permitir que a rede neural aprenda relações mais complexas entre as características latentes e a entrada original.

Um codificador e um decodificador possuem uma única camada oculta, conforme mostrado nas Equações 24 e 25. em que  $\sigma_1$  e  $\sigma_2$  são as funções de ativação da camada oculta e da camada de saída respectivamente, e o vetor  $mt$  representa as entradas mapeadas para a

dimensão latente. Os valores dos pesos  $W_1$ ,  $W_2$ ,  $b_1$  e  $b_2$  são os parâmetros a serem aprendidos pelo modelo durante a fase do treinamento (WANG et al., 2023).

$$m_t = \sigma_1(w_1x_t + b_1) \quad (24)$$

$$\hat{x}_t = \sigma_2(w_2m_t + b_2) \quad (25)$$

O procedimento de aprendizado do modelo implica na minimização da discrepância entre as entradas e saídas, formalmente representada pela variável  $J$ . A fim de mitigar os riscos de superajuste, que se manifesta quando o modelo não é capaz de generalizar além dos dados utilizados para o treinamento, emprega-se com frequência estratégias de regularização. A regularização tem por objetivo controlar a magnitude dos pesos  $W_1$  e  $W_2$  por meio da adição da soma desses pesos à função de custo (VINCENT et al., 2010; PARK et al., 2019), contribuindo assim para aprimorar o desempenho do modelo em dados não vistos. Neste trabalho é utilizada a regularização  $L_2$ , em que o termo regularizador é dado pela soma dos quadrados dos valores dos pesos, conforme apresenta a equação 26.

$$J(w_1, w_2, b_1, b_2) = \sum_i^N \frac{1}{N} \|x_t - \hat{x}_t\|^2 + \frac{\lambda}{2} (\|w_1\|^2 + \|w_2\|^2) \quad (26)$$

$\lambda$  é o coeficiente do termo regularização e define o equilíbrio entre a capacidade de generalização do modelo e o erro de treinamento.

O objetivo geral do *autoencoder* é minimizar a diferença entre a entrada original e a versão reconstruída pelo *decoder*. Para fazer isso, a rede neural usa uma função de perda que mede a diferença entre a entrada original e a reconstrução (WANG; LI; HAN, 2023). A função de perda pode ser uma medida de erro, como o erro quadrático médio (MSE) ou a divergência KL (Kullback-Leibler). A função de perda é usada para ajustar os pesos da rede neural durante o treinamento para minimizar a diferença entre a entrada original e a reconstrução (PARK et al., 2019).

Uma aplicação importante do *autoencoder* é a detecção de falhas. Essa técnica de *deep learning* pode ser treinada em dados normais e, em seguida, usado para detectar anomalias ou outliers na entrada. O *autoencoder* pode ser usado para aprender uma representação das características normais dos dados e, em seguida, identificar entradas que não correspondem a essa representação. Essa técnica é útil em diversas aplicações, como detecção de fraudes,

detecção de falhas em equipamentos e detecção de intrusos em redes de computadores. (PARK *et al.*,2019).

### 3.7 Linguagem Python

Python é uma linguagem de programação de alto nível, interpretada e orientada a objetos, que vem ganhando crescente popularidade em diversas áreas, incluindo o desenvolvimento de software, ciência de dados e inteligência artificial. A grande aceitação de Python deve-se, sobretudo, à sua simplicidade, clareza e facilidade de uso, bem como à sua vasta quantidade de bibliotecas disponíveis, que permitem a implementação rápida e eficiente de uma ampla variedade de algoritmos e técnicas (ALMEIDA *et al.*, 2018).

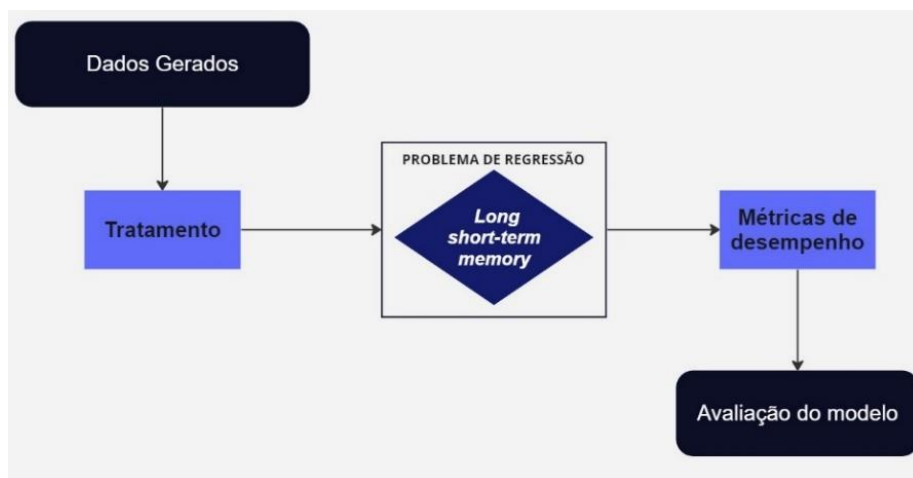
Uma das principais vantagens de Python para a aplicação de conceitos de *deep learning* é a vasta disponibilidade de bibliotecas especializadas, tais como TensorFlow, Keras e PyTorch, que fornecem uma ampla gama de ferramentas para a implementação de redes neurais profundas e outras técnicas de aprendizado de máquina. Estas bibliotecas são altamente otimizadas para o processamento de grandes volumes de dados, o que é crucial para o treinamento eficiente de modelos de *deep learning* (WALT *et al.*, 2011).

Ademais, a sintaxe simples e intuitiva de Python facilita o processo de implementação e depuração de modelos de *deep learning*, inclusive para programadores iniciantes. Python também é compatível com uma ampla variedade de plataformas e sistemas operacionais, permitindo que modelos de *deep learning* desenvolvidos em Python possam ser executados em diferentes ambientes, desde *desktops* e *laptops* até *clusters* de servidores de alta performance (ALMEIDA *et al.*, 2018).

## 4 METODOLOGIA

O presente trabalho foi realizado a partir da divisão de duas etapas. A primeira etapa do trabalho consistiu na construção dos sensores virtuais (Figura 6), que incorporam séries temporais para integração da variável tempo, capacitando a técnica utilizada a realizar previsões futuras das variáveis controladas. Nesse estágio, são estabelecidos os critérios de avaliação, utilizando a análise qualitativa fundamentada em métricas de desempenho disponíveis no conjunto de ferramentas da linguagem Python.

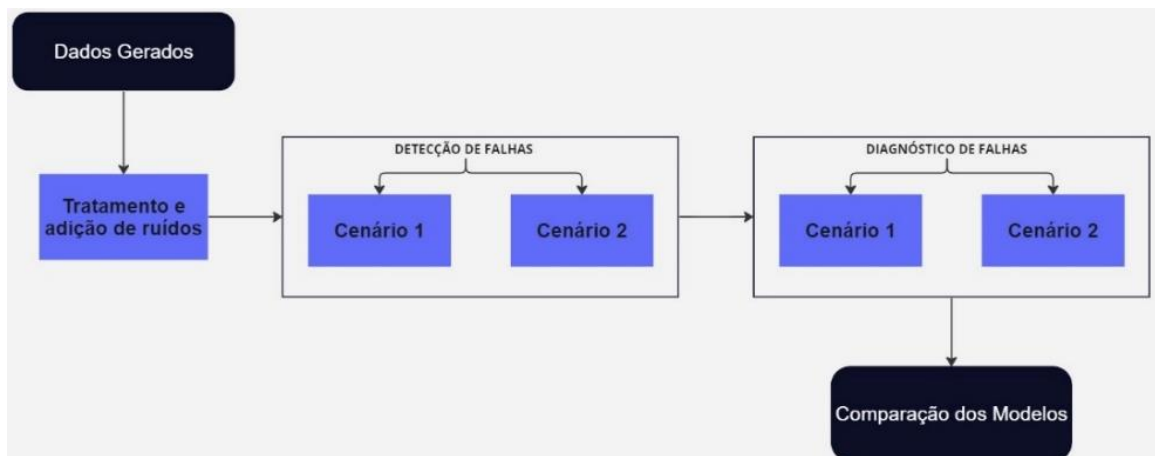
**Figura 6** - Fluxograma da 1ª parte do trabalho



**Fonte:** Elaborado pelo autor (2023)

A segunda etapa consiste na utilização das técnicas de *deep learning* na detecção e no diagnóstico de falhas, como apresenta a figura 7.

**Figura 7** - Fluxograma da 2ª parte do trabalho



**Fonte:** Elaborado pelo autor (2023)



Nessa etapa do trabalho, os cenários de detecção e diagnóstico de falhas são treinados a partir de um conjunto de dados gerados por uma rotina computacional em Python que estabelece falhas nas variáveis de processo do sistema reacional complexo, observando falhas recorrente no aspecto industrial de processos químicos. Para análises quantitativas, qualitativas e de comparação, utilizaram-se métricas de desempenho.

#### 4.1 Implementação em Python

As atividades computacionais responsáveis pelo tratamento dos dados e implementação das técnicas de aprendizado de máquina empregados nos sensores virtuais e na detecção e diagnóstico de falhas, foram desenvolvidas utilizando a linguagem de programação Python. Foi utilizado o ambiente de desenvolvimento *Spyder*, com o objetivo de simplificar tanto a criação quanto a manutenção das rotinas computacionais. As bibliotecas Pandas e Matplotlib para empregadas para análise e visualização dos dados. As ferramentas computacionais utilizadas para o desenvolvimento do presente trabalho, estão expostas no quadro 13.

**Quadro 13** - Ferramentas computacionais para criação dos códigos

<b>Linguagem de programação</b>	Python 3.10.9
<b>Bibliotecas Python</b>	Matplot 3.7.0
	Scikit-learn - 1.2.2
	Pandas 1.5.3
	Tensorflow 2.12.0
<b>Editor do código</b>	Spyder 5.4.3

**Fonte:** Elaborado pelo autor (2023)

#### 4.2 Sensor virtual

Os conjuntos de dados foram divididos em dois: 80% para treinamento e 20% para validação, carregados de forma separada e normalizados de maneira similar para garantir a consistência dos dados. Os conjuntos de dados foram utilizados para criação, treinamento e validação dos modelos de sensores virtuais.

### 4.2.1 Tratamento dos dados

A biblioteca *sklearn* é utilizada para normalizar os dados de treinamento e validação. A normalização é importante para garantir que todas as características estejam na mesma escala. Os dados normalizados são redimensionados para o formato esperado pela camada *LSTM* do modelo.

A normalização dos dados desempenha um papel crucial nas simulações dos sensores virtuais com a técnica *LSTM*. Essa etapa tem como objetivo preparar os dados de entrada para o modelo *LSTM*, garantindo um melhor desempenho e resultados mais precisos. A importância da normalização dos dados reside no fato de que diferentes características (ou variáveis) dos dados podem ter escalas e unidades de medida distintas. Ao normalizar os dados, ou seja, ajustá-los para uma escala comum, elimina-se a influência das diferenças de escala entre as variáveis. Isso é particularmente relevante na técnica *LSTM*, que é sensível a variações de magnitude dos dados de entrada. A normalização permite que o modelo aprenda de forma equilibrada a partir de todas as características dos dados, evitando que algumas variáveis dominem as outras devido a diferenças de escala.

### 4.2.2 Criação e treinamento do modelo

O modelo foi criado utilizando a biblioteca *Keras*. Foram adicionadas uma camada *LSTM* com 16 unidades de memória e função de ativação *ReLU*, uma camada *Dense* com 8 unidades e função de ativação *ReLU* e uma camada *Dense* com 1 unidade (saída do modelo).

O modelo foi compilado com o otimizador "adam" e a função de perda *MSE* (erro quadrático médio). O Adam (Adaptive Moment Estimation) é um otimizador popularmente utilizado em algoritmos de aprendizado de máquina, especialmente em redes neurais profundas (KINGMA et al., 2014). O modelo é treinado utilizando os dados de treinamento e validação, com um número máximo de épocas (100) e um tamanho de lote de 32. Durante o treinamento, as métricas são monitoradas e a parada antecipada é ativada caso a perda no conjunto de validação não melhore por 10 épocas consecutivas.

### 4.2.3 Avaliação do desempenho

O modelo é utilizado para fazer previsões no conjunto de validação e utilizou-se as métricas de regressão disponíveis na biblioteca; sendo o *R2*, o erro absoluto médio (*Mean*

*Absolute Error* – MAE) e o erro quadrático médio, MSE (da sigla em inglês *Mean Squared Error*). No geral, o código carrega os dados de treinamento e validação, normaliza-os, constrói um modelo LSTM, treina o modelo, avalia seu desempenho e gera um gráfico de dispersão. O modelo treinado pode ser salvo e carregado posteriormente para fazer previsões em novos dados.

### 4.3 Detecção e Diagnóstico de Falhas (FDD)

A metodologia foi constituída inicialmente por uma revisão sistemática da literatura acerca do Controle Estatístico de Processos, de estratégias de *FDD* e das técnicas de *deep learning* no que diz respeito às Redes Neurais *Autoencoder* e *LSTM*.

O conjunto de dados com melhor desempenho no desenvolvimento dos sensores virtuais foi utilizado para a inserção de falhas que simulam erros no funcionamento dos sensores físicos para o treinamento e teste dos algoritmos de detecção e diagnóstico de falhas, 80% desses dados foram utilizados para treinamento e 20% utilizados para teste. Foram definidos 6 tipos de falhas randômicas com variação de -90% a +90% como apresenta o quadro 14.

**Quadro 14** - Falhas aplicadas às variáveis do processo

Variável	Falha do sensor	Diagnóstico
-	-	Normal
$C_{A0}$	[-90%; +90%]	Falha 1
$T_{r0}$	[-90%; +90%]	Falha 2
$Q_c$	[-90%; +90%]	Falha 3
$q_r$	[-90%; +90%]	Falha 4
$T_r$	[-90%; +90%]	Falha 5
$T_c$	[-90%; +90%]	Falha 6

**Fonte:** Elaborado pelo autor (2023)

A tabela 2 apresenta como os dados foram organizados com a inserção das falhas randômicas nas variáveis do processo.

**Tabela 2** - Dados de processos após a inserção de falhas

Tr (K)	Tc (K)	Tr0 (K)	CA0 (kmol/m <sup>3</sup> )	Qc (kJ/min)	qr (m <sup>3</sup> /min)	Classes_Falhas
391.3895074	401.3905	385.4551	4.7246	144.5	0.002904	normal
391.3130398	401.3221	385.4551	5.01026925	144.5	0.002904	Falha_1
391.4754546	401.4014	684.5013972	4.7246	144.5	0.002904	Falha_2
391.4337154	401.3711	385.4551	4.7246	244.6217379	0.002904	Falha_3
391.3750282	401.3369	385.4551	4.7246	144.5	0.001822607	Falha_4
740.1593199	401.3862	385.4551	4.7246	144.5	0.002904	Falha_5
391.3201121	750.8433	385.4551	4.7246	144.5	0.002904	Falha_6

**Fonte:** Elaborado pelo autor (2023)

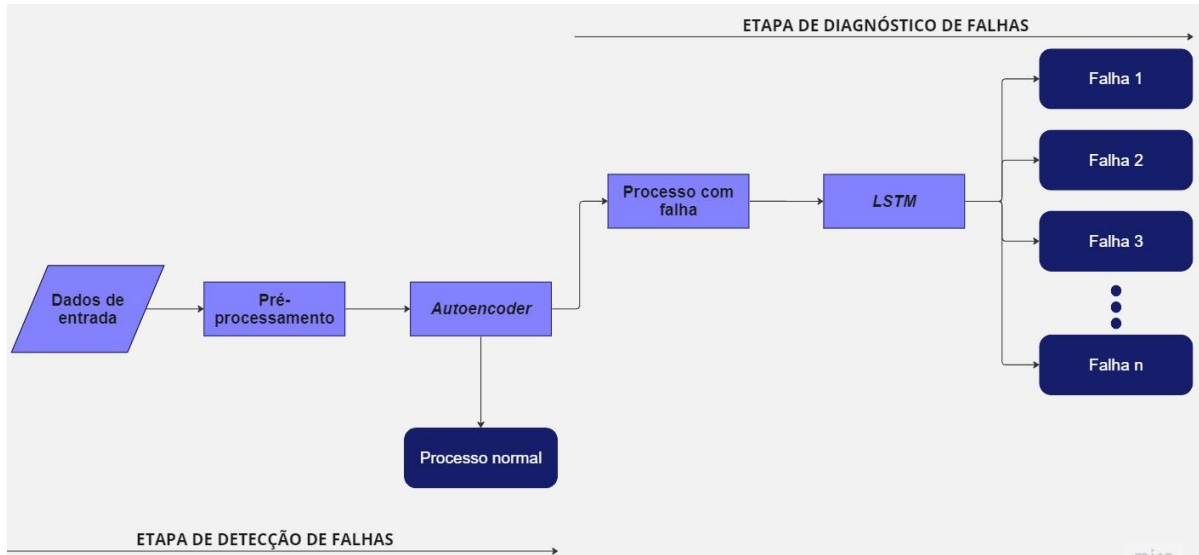
Após o tratamento dos dados, foi implementada as etapas de detecção e de diagnóstico de falhas. Duas estratégias distintas, denominadas Cenário 1 e Cenário 2. No Cenário 1, representado pela Figura 8, a detecção e o diagnóstico de falhas ocorrem simultaneamente em uma única etapa de processamento, esse cenário foi proposto por Soares (2017). Nesse sentido, os dados provenientes de operações normais e falhas são alimentados como entrada em uma Rede Neural do tipo Long Short-Term Memory (LSTM), sendo tratados como um problema de classificação.

**Figura 8** - Cenário 1 para a detecção e diagnóstico de falhas

**Fonte:** Elaborado pelo autor (2023)

No Cenário 2, representado pela Figura 11, a detecção de falhas é realizada de forma independente pelo *Autoencoder*, e na etapa de diagnóstico, apenas os dados de operação com falhas são empregados como entrada nos algoritmos de aprendizado supervisionado.

**Figura 9** - Cenário 2 para a detecção e diagnóstico de falhas



**Fonte:** Elaborado pelo autor (2023)

Com base em tais cenários, este estudo objetivou avaliar o desempenho de ambas as estratégias, determinar qual delas apresentou melhor performance e investigar os fatores que exerceram influência sobre o desempenho de cada uma delas.

Para a detecção e diagnóstico de falhas, foram utilizadas as seguintes métricas de classificação para avaliação de desempenho: Acurácia, f1 – Score, índice de Jaccard, Recall, precisão e taxa de erro.

## 4.4 Métricas de desempenho

### 4.4.1 Erro Médio Absoluto - MAE

O erro médio absoluto (MAE) é uma medida de avaliação utilizada em modelos de aprendizado de máquina, calculado a partir da média dos erros absolutos. O cálculo do MAE utiliza o módulo de cada erro para evitar a subestimação dos resultados, tornando a medida menos afetada por valores extremos (outliers) (GERON, 2019).

Cada erro é definido como a diferença entre o valor observado ( $Y$ ) e o valor previsto pelo modelo ( $\hat{Y}$ ). O MAE é calculado somando os módulos das diferenças entre  $Y$  e  $\hat{Y}$  para

cada registro na amostra, e dividindo esse somatório pelo número total de registros na amostra. A equação 27 apresenta o cálculo para definição do MAE.

$$MAE = \frac{1}{n} \sum_{i=1}^n |Y_i - \hat{Y}_i| \quad (27)$$

#### 4.4.2 Erro Quadrático Médio – MSE

O erro quadrático médio (MSE) é uma medida comumente utilizada para avaliar a acurácia de modelos de aprendizado de máquina. O MSE é calculado elevando ao quadrado cada erro individual e, em seguida, calculando a média desses erros quadráticos. Elevar ao quadrado cada erro, o MSE dá um maior peso aos maiores erros, o que pode ser útil em alguns casos. No entanto, é importante destacar que a utilização do MSE pode tornar o modelo mais sensível a valores extremos (*outliers*) e pode não ser apropriado para todas as situações de modelagem (GERON, 2019). A equação 28 apresenta o cálculo para definição do MSE.

$$MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2 \quad (28)$$

#### 4.4.3 Índice de Jaccard

O Índice de Jaccard, também conhecido como Coeficiente de Jaccard ou Similaridade de Jaccard, é uma métrica que mede a sobreposição entre duas amostras. Ele é calculado pela divisão do tamanho da interseção das amostras pelo tamanho da união das amostras (CHEN et al., 2008). A equação 30 apresenta o cálculo do índice de Jaccard:

$$S_j = \frac{a}{a + b + c} \quad (29)$$

A equação para o cálculo do Índice de Jaccard é baseada em três variáveis: "a", "b" e "c". O valor de "a" representa o número de espécies que são encontradas em ambos os locais (A e B). O valor de "b" representa o número de espécies encontradas no local B, mas que não são encontradas em A. Já o valor de "c" representa o número de espécies encontradas no local A, mas que não são encontradas em B.

#### 4.4.4 Matriz de confusão

A matriz de confusão é uma ferramenta importante na análise de desempenho de modelos de classificação. Ela é uma tabela que mostra a frequência com que as classes do modelo são previstas corretamente ou incorretamente. A matriz de confusão é formada por quatro células que representam os resultados do modelo em relação às classes reais. Essas células são: verdadeiro positivo (VP), falso positivo (FP), verdadeiro negativo (VN) e falso negativo (FN) (GERON, 2019). A figura 10 apresenta uma matriz de confusão.

**Figura 10** - Matriz de confusão

		Valor Predito	
		Sim	Não
REAL	Sim	Verdadeiro positivo (VP)	Falso Negativo (FN)
	Não	Falso Positivo (FP)	Verdadeiro negativo (VP)

**Fonte:** Elaborado pelo autor (2023)

- **Verdadeiro positivo (VP):** ocorre quando o modelo prevê corretamente uma instância da classe positiva.
- **Falso positivo (FP):** ocorre quando o modelo prevê erroneamente uma instância como positiva quando na verdade é negativa.
- **Verdadeiro negativo (VN):** ocorre quando o modelo prevê corretamente uma instância da classe negativa;
- **Falso negativo (FN):** ocorre quando o modelo prevê erroneamente uma instância como negativa quando na verdade é positiva.

A partir da matriz de confusão é possível avaliar o desempenho de algoritmos a partir das métricas de classificação sendo elas: Acurácia, f1-score, precisão e recall. É possível observar as métricas e suas equações 30, 31, 32 e 33 no quadro 15.

**Quadro 15** - Métricas de desempenho de classificação

<b>Métrica</b>	<b>Fórmula</b>
Acurácia	$Acc = \frac{VP+VN}{VP+VN+FP+FN} \quad (30)$
F1-Score	$Fs = 2 \cdot \frac{Pre \cdot Recall}{Prec + Recall} \quad (31)$
Precisão	$Prec = \frac{VP}{FP+VP} \quad (32)$
Recall	$Recall = \frac{VP}{FN+VP} \quad (33)$

**Fonte:** adaptado de Geron (2019)

## 4.5 Apresentação dos resultados

A apresentação dos resultados teve início com a exposição dos dados obtidos por meio dos sensores virtuais desenvolvidos. Esses dados representaram informações do ajuste dos sensores em relação aos tempos de amostragem e tempos de atraso. A organização dos dados permitiu uma análise mais detalhada e abrangente dos resultados alcançados.

Em seguida, os resultados foram analisados com base em métricas de desempenho amplamente utilizadas na avaliação de modelos de aprendizagem de máquina, o erro Quadrático médio (MSE), erro absoluto médio (MAE) e coeficiente de determinação (R<sup>2</sup>). Essas métricas forneceram informações valiosas sobre a precisão e a qualidade das previsões realizadas pelos modelos.

Posteriormente, foram apresentados os resultados das estratégias de detecção de falhas implementadas nos cenários 1 e 2. Os resultados obtidos nessas abordagens foram discutidos e comparados, destacando-se as diferenças de desempenho e eficácia entre elas. Após essa análise, foram realizadas avaliações adicionais utilizando as métricas de desempenho Acurácia, F1-score, Recall, Índice de Jaccard e taxas de erro. Essas métricas forneceram uma visão mais abrangente sobre a capacidade dos modelos em detectar e classificar corretamente as falhas ocorridas no processo químico. A discussão dessas métricas permitiu compreender a efetividade das estratégias implementadas e a sua capacidade de lidar com diferentes tipos de falhas.

Em suma, a apresentação dos resultados contemplou desde a exposição dos dados dos sensores virtuais até a análise detalhada das métricas de desempenho, proporcionando uma



visão abrangente e fundamentada sobre o desempenho das estratégias de detecção e diagnóstico de falhas. Esses resultados foram fundamentais para a validação e aprimoramento dos modelos desenvolvidos.

## 5 RESULTADOS E DISCUSSÃO

A fim de apresentar e interpretar adequadamente os resultados, optou-se por uma abordagem sistemática e metodológica, na qual os dados foram organizados e comentados de forma simultânea, formando uma linha de raciocínio interdependente e coerente. Dessa forma, busca-se garantir a confiabilidade e a precisão dos resultados e das considerações formuladas a partir deles.

### 5.1 Sensor virtual

Para cada conjunto de dados, foram calculadas as métricas de desempenho utilizando a combinação de hiper parâmetros que obteve os maiores valores de  $R^2$  após otimização para as redes neurais *LSTM*. A tabela 3 apresenta o desempenho dos sensores virtuais em tempos de amostragem e tempos de atraso diferentes

**Tabela 3** - Desempenho dos sensores virtuais em tempos de amostragem e tempos de atraso distintos

Dados	MSE x 10 <sup>7</sup>	R <sup>2</sup>	MAE x 10 <sup>4</sup>
3 segundos e 1 tempo de atraso	0,49	0,999973	1,34
3 segundos e 2 tempos de atraso	0,03	0,999998	0,44
3 segundos e 3 tempos de atraso	0,52	0,999971	0,22
6 segundos e 1 tempo de atraso	2,14	0,999884	3,38
6 segundos e 2 tempos de atraso	0,63	0,999965	2,23
6 segundos e 3 tempos de atraso	0,62	0,999966	2,37
9 segundos e 1 tempo de atraso	2,84	0,999846	3,35
9 segundos e 2 tempos de atraso	1,51	0,999918	3,42
9 segundos e 3 tempos de atraso	0,03	0,999998	0,50
12 segundos e 1 tempo de atraso	10,93	0,999407	8,54
12 segundos e 2 tempos de atraso	1,00	0,999946	1,10
12 segundos e 3 tempos de atraso	11,39	0,999383	10,49

**Fonte:** Elaborado pelo autor (2023)

Dentre os tempos de amostragem, observou-se que mesmo com o aumento da amostragem, os valores de  $R^2$  não foram comprometidos e permaneceram elevados,  $R^2$  em

todos os conjuntos de dados obtiveram valores superiores a 0,999. Já os valores de MSE Variaram entre  $0,03 \times 10^{-7}$  a  $11,03 \times 10^{-7}$ , esses resultados demonstram o ajuste dos sensores virtuais em uma diferença de erro entre o valor predito e o real próximos de zero. Da mesma forma é possível observar com os valores de MAE, com variação entre  $0,22 \times 10^{-4}$  a  $10,49 \times 10^{-4}$ .

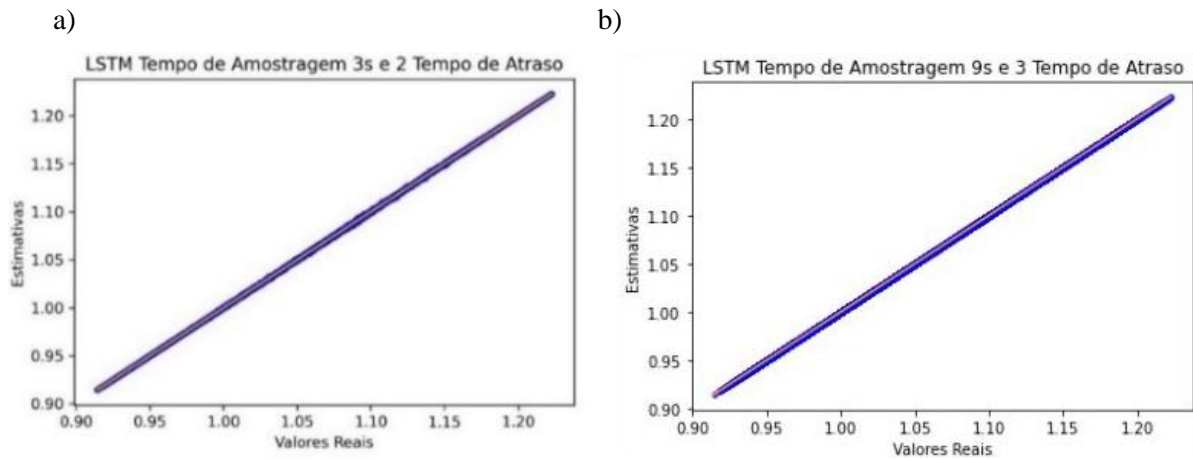
Analisando os conjuntos de dados de 3 segundos de amostragem, o conjunto com 2 tempos de atrasos obteve melhor performance que 1 tempo de atraso e 3 tempos de atraso, tendo em vista o maior valor de  $R^2$ . Ademais, o valor de MSE do conjunto de 3 tempos de atraso é significativamente menor, corroborando a superioridade da performance deste conjunto em relação aos demais. Por sua vez, os conjuntos de 1 e 3 tempos de atrasos apresentam valores elevados de MAE, em comparação com o conjunto de 2 tempos de atraso.

Na análise do conjunto de dados o tempo de amostragem de 6 segundos, foram avaliadas as performances dos conjuntos e verificou-se que os conjuntos de 2 e 3 tempos de atrasos apresentaram desempenhos semelhantes, embora o conjunto com 2 tempos de atraso tenha apresentado um valor de MSE menor, o que indica uma menor taxa de erro na previsão. O conjunto de dados de 1 tempo de atraso, por outro lado, obteve resultados inferiores nas métricas analisadas. O conjunto de dados de 9 segundos de amostragem, evidenciou a performance superior do conjunto com 3 tempos de atraso com melhores desempenho nas métricas. A performance do conjunto de dados de 12 segundos de amostragem se mostrou elevada com 2 tempos de atraso.

Os melhores conjuntos de dados analisados neste trabalho, foram 9 segundos de amostragem com 3 tempos de atraso com  $R^2$  0,999998, MSE de  $0,03 \times 10^{-7}$  e também MAE de  $0,50 \times 10^{-4}$ , e o conjunto de dados de 3 segundos de amostragem com 2 tempos de atraso com  $R^2$  0,999998, MSE de  $0,03 \times 10^{-7}$  e MAE de  $0,44 \times 10^{-4}$ . Apesar dos dados utilizados serem provenientes de simulação matemática com dados padronizados, esses valores de desempenho evidenciaram a eficácia das redes neurais LSTM em realizar a inferência estatística.

Para a análise visual e corroboração dos resultados, foi utilizado o gráfico de paridade, a fim de avaliar a qualidade do modelo de inferência estatística. o gráfico de paridade é uma ferramenta valiosa na análise de algoritmos de inferência estatística, pois fornece uma representação visual dos dados e ajuda a entender a relação entre as variáveis envolvidas. Ele auxilia na avaliação da qualidade do modelo, na identificação de outliers, no reconhecimento de tendências e na detecção de possíveis vieses. A figura 11 apresenta o gráfico de paridade dos dois melhores conjuntos de dados.

**Figura 11** - Gráficos de paridade dos sensores virtuais com melhores tempos de amostragem e tempos de atraso



**Fonte:** Elaborado pelo autor (2023)

A figura 11-a) representa a performance do sensor virtual com o conjunto de 3 segundos de amostragem com 2 tempos de atraso, já a figura 11-b) é apresentado o sensor virtual do conjunto de dados de 9 segundos de amostragem e 3 tempos de atrasos. Ambos os gráficos evidenciam o ajuste dos sensores virtuais à linha de tendência, e a captação correta da relação das variáveis do processo.

## 5.2 Detecção e Diagnóstico de Falhas

Em contraste com os dados destinados ao sensor virtual, foi executado apenas o processo de padronização dos dados, baseando-se que a ocorrência de falhas e sua detecção ocorrem simultaneamente no mesmo instante de tempo. O conjunto de dados utilizado para inserção de falhas foi o de 9 segundos de amostragem.

Apenas a padronização dos dados de entrada foi realizada, visando estabelecer uma média zero e uma variância unitária. Por outro lado, o conjunto de saída não exigiu padronização, pois consiste nos rótulos das classes. Assim como no caso do sensor virtual, utilizando a biblioteca *Scikit-learn*, o conjunto de dados foi dividido em dois subconjuntos: "Treino", composto por 80% das amostras, e "Teste", contendo 20% das amostras. Além disso, o treinamento dos modelos foi realizado com os 6 tipos de falhas diferentes mais a classe correspondente ao status de operação normal, totalizando 7 entradas nas redes neurais, e foram calculadas as métricas de desempenho na etapa de teste

Considerando que um dos propósitos deste estudo foi investigar o processo de detecção e diagnóstico de falhas empregando diferentes técnicas de *deep learning*, os resultados foram

organizados por cada um dos cenários delineados na metodologia e, posteriormente, o desempenho de cada abordagem foi comparado e debatido.

### 5.2.1 Cenário 1- Detecção e Diagnóstico de Falhas na mesma etapa

Para este cenário, a rede neural *LSTM* foi treinada para a tarefa de classificação de falhas. O modelo foi treinado por 50 épocas, em que cada época representa uma passagem completa pelos dados de treinamento. Além disso, a função de ativação utilizada foi a *softmax*, essa função é comumente utilizada em problemas de classificação multiclasse para produzir probabilidades de pertencimento a cada classe. O modelo possui duas camadas ocultas, sendo cada uma delas composta por 50 neurônios. A avaliação do desempenho foi realizada a partir da matriz de confusão apresentada na figura 12.

**Figura 12** - Matriz de confusão da FDD utilizando *LSTM*

Matriz de Confusão

Classe verdadeira \ Classe prevista	Falha_1	Falha_2	Falha_3	Falha_4	Falha_5	Falha_6	normal
Falha_1	92127	12	118	249	0	3	7465
Falha_2	60	97839	51	66	22	23	1576
Falha_3	115	25	95437	114	5	8	4089
Falha_4	226	13	89	91210	0	0	7426
Falha_5	5	29	23	11	99114	300	201
Falha_6	8	21	35	2	251	99236	197
normal	1097	579	1301	1116	24	33	594278

**Fonte:** Elaborado pelo auto (2023)

A partir da matriz de confusão, tornou-se possível avaliar o desempenho por falha do processo, os valores estão expressos na tabela 4.

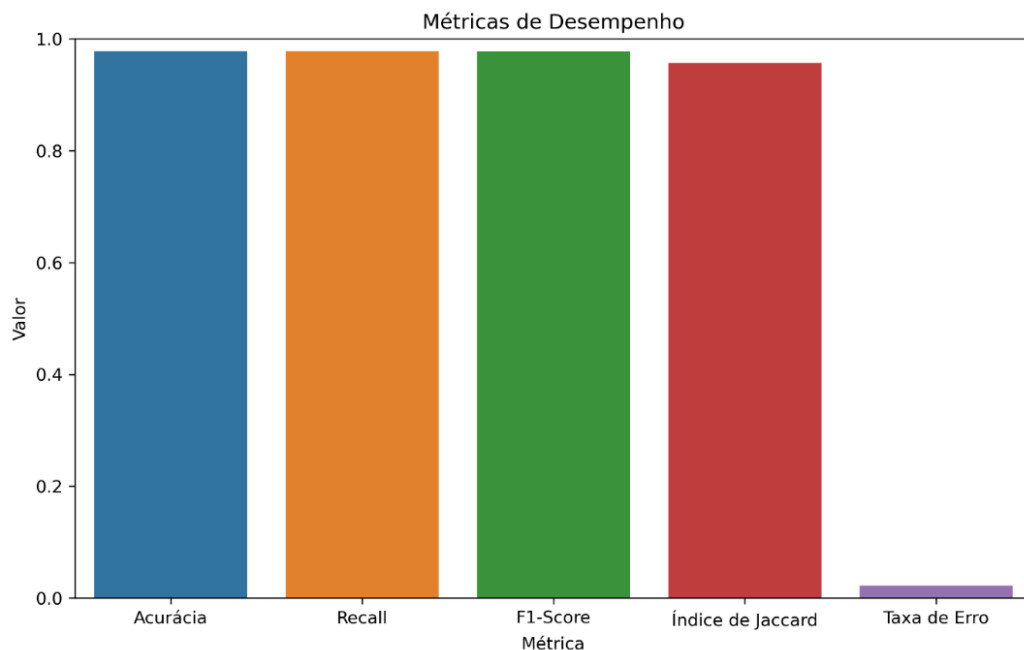
**Tabela 4** - Avaliação de métricas por falha do processo do cenário 1

<b>Avaliação de métricas por falha do processo</b>			
<b>Tipo</b>	<b>Precisão</b>	<b>Recall</b>	<b>F1-score</b>
Processo normal	0,98	0,92	0,95
Falha 1	0,99	0,98	0,99
Falha 2	0,98	0,96	0,97
Falha 3	0,98	0,92	0,95
Falha 4	1,00	0,99	1,00
Falha 5	1,00	0,99	1,00
Falha 6	0,97	0,99	0,98

**Fonte:** Elaborado pelo auto (2023)

Nota-se que o modelo obteve uma performance satisfatória, visto que todas as análises dos tipos de processo garantiram precisão igual ou superior a 97%, além disso, os valores de Recall também atingiram valores consideráveis, indicando que o modelo está identificando corretamente uma maior proporção de valores positivos das classes.

Ao analisar o modelo de maneira geral, é possível notar a eficiência a partir das métricas de desempenho que estão expressas na figura 13 e na tabela 5.

**Figura 13** – Gráfico das métricas de desempenho para o cenário 1

**Fonte:** Elaborado pelo auto (2023)

**Tabela 5** - Avaliação do desempenho do Cenário 1

<b>Avaliação do Cenário 1</b>	
<b>Métrica</b>	<b>Desempenho</b>
Acurácia	98%
Recall	98%
F1-Score	98%
Índice de jaccard	96%
Taxa de Erro	2%
Taxa de erros na detecção	3,90%
Taxa de erros no diagnóstico	1,18%

**Fonte:** Elaborado pelo auto (2023)

As métricas, acurácia, recall e f1 – score, registraram o desempenho de 98%. A acurácia alta indica que o modelo está fazendo previsões corretas na maioria dos casos, o que é um indicador positivo, além disso, valores altos de acurácia aumentam a confiança nas previsões feitas pelo modelo. Um F1-Score alto é útil, pois considera tanto a capacidade de identificar corretamente exemplos positivos (recall) quanto a capacidade de evitar classificações incorretas (precisão). Já o recall alto indica a capacidade do modelo de identificar corretamente os valores positivos das classes de interesse. O Índice de Jaccard registrou o desempenho de 96%, considerado alto, indicando que o modelo tem uma alta taxa de concordância com os exemplos verdadeiros positivos, o que é um indicador positivo de desempenho. Corroborando com as demais métricas, a taxa de erro do modelo atingiu o valor de 2%, considerado um erro pequeno e aceitável.

Na tabela 5, é possível observar a taxa de erro do modelo, tanto na detecção quanto no diagnóstico. A detecção do modelo apresenta um erro de 3,90%, indicando que o modelo tem uma baixa taxa de casos positivos não detectados ou negligenciados. Já no diagnóstico das falhas, o modelo obteve uma taxa de erro de 1,18%, e essa taxa está diretamente relacionada à capacidade do modelo de diagnosticar corretamente as falhas. Uma taxa de erro baixa significa que o modelo tem uma baixa taxa de casos de falhas não detectadas, permitindo uma detecção mais precoce e, portanto, intervenções rápidas para corrigir as falhas antes que causem danos ou interrupções graves no sistema.

Falhas não detectadas ou diagnósticos incorretos podem levar a paradas não programadas e interrupções indesejadas nos processos industriais. Uma taxa de erro baixa

contribuiu para a minimização dessas paradas, permitindo uma identificação precisa e oportuna das falhas, o que, por sua vez, ajuda a evitar a perda de produção e reduzir os custos associados.

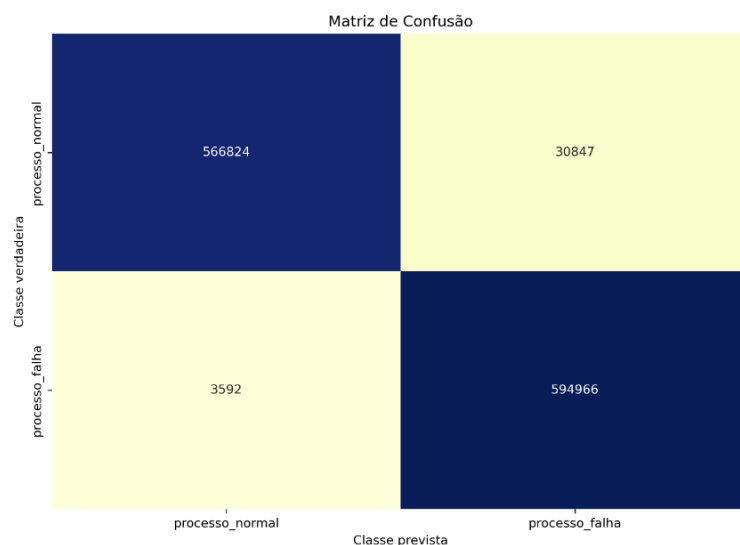
## 5.2.2 Cenário 2- Detecção e Diagnóstico de Falhas em etapas separada

Neste cenário, foram utilizadas duas técnicas de *deep learning*, o Autoencoder para a etapa de detecção, e o *LSTM* para a etapa de diagnóstico de falhas, assim como o trabalho proposto por Park et al (2019).

### 5.2.2.1 Detecção de Falhas

O Autoencoder foi treinado utilizando uma camada de entrada, uma camada oculta (*encoder*) e uma camada de saída (*decoder*). O número de neurônios na camada oculta foi definido como 32. Já a função de ativação utilizada foi a ReLU para a camada do *encoder* e *softmax* para a camada do *decoder*. O otimizador utilizado foi o Adam com uma taxa de aprendizagem de 0.001. A função de perda utilizada foi o MSE. O *Autoencoder* é treinado por 100 épocas, com um tamanho de lote de 32 e uma proporção de validação de 0.2. A avaliação do desempenho foi realizada a partir da matriz de confusão apresentada na figura 14.

**Figura 14** - Matriz de confusão da etapa de detecção de falhas do cenário 2



**Fonte:** Elaborado pelo auto (2023)

A partir da matriz de confusão, tornou-se possível avaliar o desempenho por processo, os valores estão expressos na tabela 6.



**Tabela 6** - Desempenho por processo da etapa de detecção de falhas - Cenário 2

<b>Desempenho por processo</b>			
<b>Tipo de processo</b>	<b>Precisão</b>	<b>Recall</b>	<b>F1-Score</b>
Processo Normal	0,99	0,95	0,97
Processo com falha	0,95	0,99	0,97

**Fonte:** Elaborado pelo auto (2023)

Nota-se que a precisão da detecção do processo normal foi elevada, evidenciando a eficiência do modelo. Já o Recall elevado do processo com falhas, indicando a eficiência do modelo em detectar as falhas do processo. Além disso, o f1 – score com 97% reflete um modelo que é capaz de alcançar tanto alta precisão quanto alto recall, o que é valioso em muitos cenários de tomada de decisão.

A tabela 7 apresenta o desempenho geral da etapa de detecção de falhas utilizando o Autoencoder.

**Tabela 7** - Desempenho geral da etapa de detecção de falhas - Cenário 2

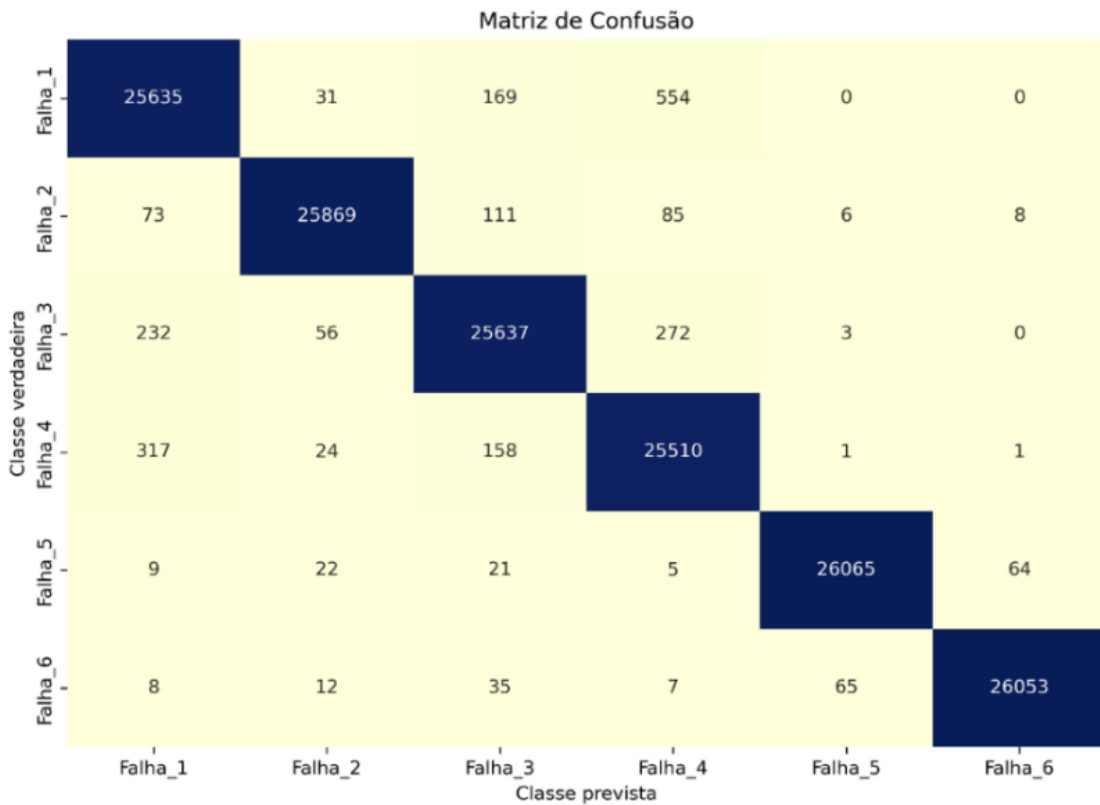
<b>Desempenho geral da etapa de detecção -cenário 2</b>	
<b>Métrica</b>	<b>Porcentagem</b>
Acurácia	97%
Taxa de Erro nas previsões	2,88%

**Fonte:** Elaborado pelo auto (2023)

A acurácia de 97% evidencia o desempenho de um modelo de classificação aceitável. Na análise da taxa de erro nas previsões, que está diretamente relacionado na capacidade do modelo detectar falhas, o valor da taxa de erro 2,88% apresentou um modelo confiável e com bom desempenho nessa etapa de detecção.

### 5.2.2.2 Diagnóstico de falhas

Para esta etapa, a arquitetura e os parâmetros da técnica de *deep learning LSTM* foram os mesmos utilizados para o cenário 1. No entanto, no cenário 2, foram realizados o treinamento e o teste do modelo considerando apenas os dados dos 6 tipos de falhas, sem incluir os dados de operação normal. A performance do modelo para o cenário 2 está expressa na figura 15.

**Figura 15** - Matriz de confusão do diagnóstico de falhas - Cenários 2

**Fonte:** Autor (2023)

A partir da matriz de confusão, tornou-se possível avaliar o desempenho por falha do processo, os valores estão expressos na tabela 8.

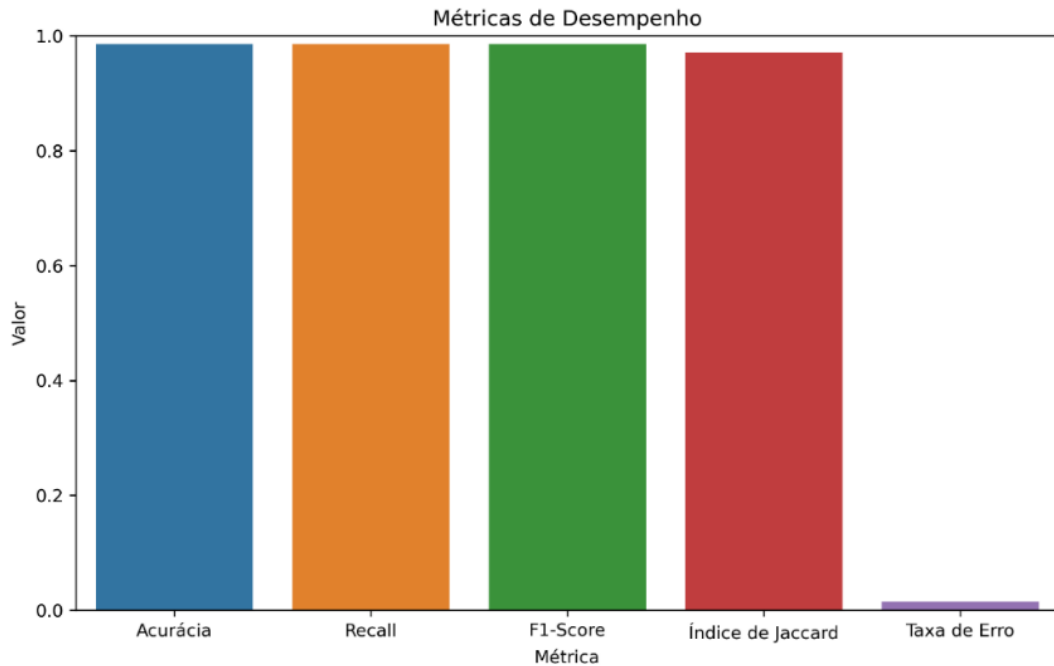
**Tabela 8** - Avaliação de métricas por falha do processo de diagnóstico de falhas - Cenário 2

<b>Avaliação de métricas por falha do processo</b>			
<b>Tipo</b>	<b>Precisão</b>	<b>Recall</b>	<b>F1-score</b>
Falha 1	0,98	0,97	0,97
Falha 2	0,99	0,99	0,99
Falha 3	0,98	0,98	0,98
Falha 4	0,97	0,98	0,97
Falha 5	1,00	1,00	1,00
Falha 6	1,00	1,00	1,00

**Fonte:** Autor (2023)

A partir da análise por falha do processo, é possível notar o alto desempenho do modelo em nessa etapa do processo. Na figura 16 estão expressas as informações das métricas gerais do desempenho da etapa de diagnóstico de falhas.

**Figura 16** - Gráfico do desempenho da etapa de diagnóstico de falhas - Cenário 2



Fonte: Autor (2023)

Na tabela 9 é possível identificar os altos valores das métricas de desempenho. A taxa de erro de 1% e a acurácia de 99% evidencia a eficácia e confiabilidade do modelo em diagnosticar falhas.

**Tabela 9** - Avaliação geral da etapa de diagnóstico de falhas - Cenário 2

Métrica	Porcentagem
Taxa de Erro nas previsões	1%
Acurácia	99%
F1 – Score	99%
Recall	99%
Indice de Jaccard	97%

Fonte: Autor (2023)

### 5.2.3 Comparação dos cenários

A tabela 10 apresenta as performances dos dois cenários na etapa de diagnóstico de falhas.

**Tabela 10** - Comparação do desempenho da etapa de diagnóstico de falhas

<b>Desempenho dos cenários na Etapa de diagnóstico de falhas</b>		
<b>Cenário</b>	<b>Acurácia</b>	<b>Erro</b>
Cenário 1	98%	1,18%
Cenário 2	99%	1,02%

**Fonte:** Autor (2023)

Pode-se observar na tabela 10 que mesmo sendo valores próximos, houve uma melhoria na acurácia e Taxa de erro. Isso se deve pelo fato de a etapa de diagnóstico trabalhar com uma quantidade menor de valores, visto que os dados de operação normal não foram inseridos no treinamento e, do ponto de vista do problema de classificação, o modelo ficou mais simples por ter menos classes para identificar. Nota-se que a *LSTM* no cenário 2, apenas com dados de falha, performou melhor para todas as classes em comparação com o cenário 1.

## 6 CONCLUSÃO

Com base nos resultados obtidos, pode-se afirmar que o trabalho atingiu os objetivos desejados, com resultados satisfatórios e significativos dentro do escopo do trabalho. Implementou-se, em linguagem de programação Python, o desenvolvimento de sensores virtuais e estratégias de detecção e diagnóstico de falhas em um processo reacional complexo, utilizando as técnicas de *deep learning Autoencoder* e *LSTM*.

De forma geral, constatou-se que as Redes Neurais *LSTM* demonstraram uma estrutura eficaz para lidar com dados sequenciais. Com base nas métricas de desempenho do sensor virtual, foi constatado que a redução do intervalo de amostragem resulta em melhorias na precisão da previsão do sensor. No entanto, essa diminuição está associada a um aumento significativo no número de amostras, o que gera um aumento exponencial na carga computacional requerida.

No contexto da detecção e diagnóstico de falhas, constatou-se que as métricas utilizadas demonstraram resultados notáveis, apesar dos dados provenientes de simulação matemática, evidenciando eficácia nas etapas de detecção e diagnóstico. Esses resultados foram observados mesmo diante da ocorrência simultânea de múltiplas falhas, em diferentes amplitudes. A utilização do acoplamento entre a rede Autoencoder e a *LSTM*, conforme proposto no cenário 2, resultou em um desempenho superior em relação à abordagem mais tradicional observada no cenário 1. Essa abordagem híbrida revelou-se de grande relevância prática, especialmente considerando que dados provenientes de ambientes industriais tendem a apresentar um volume significativamente maior de dados normais de operação em comparação com o volume de dados contendo falhas.

Em suma, os resultados obtidos nas métricas de detecção e diagnóstico de falhas demonstraram um desempenho excelente, evidenciando a robustez das técnicas utilizadas e a capacidade de lidar com cenários complexos. Esses resultados reforçam a importância de uma abordagem adequada para a detecção de falhas em sistemas críticos, possibilitando ações oportunas e eficazes para minimizar os impactos negativos e garantir a continuidade operacional. Para trabalhos futuros é indicado a avaliação dessas técnicas com dados de processos reais e consolidar a inteligência artificial no ambiente industrial.

Durante a execução deste trabalho, a linguagem de programação Python mostrou-se altamente robusta na implementação das técnicas de aprendizado de máquina. Todo o trabalho foi realizado com sucesso dentro dos recursos disponibilizados pela linguagem, sem a necessidade do uso de softwares externos. Este fato ressalta a importância das técnicas de *deep*

*learning* como ferramentas robustas e imprescindíveis na formação do engenheiro químico, uma vez que a indústria 4.0 está cada vez mais disseminada. Essas técnicas viabilizam previsões precisas e contínuas sobre um processo, em tempo real, com custos de implementação próximos a zero, desde que adequadamente treinadas.

Por fim, esta experiência propiciou a aquisição de conhecimentos e aprimoramento em técnicas e conceitos que geralmente não são abordados de maneira aprofundada durante a graduação. Isso permitiu a exploração de tópicos interligados às disciplinas de modelagem e simulação, métodos numéricos, cinética de reatores e diversas outras áreas de conhecimento que contribuíram significativamente. Por meio dessa abordagem, foi possível aprofundar a compreensão e aplicação prática de conceitos relacionados a essas disciplinas, ampliando o conhecimento sobre processos químicos, suas simulações, as técnicas numéricas envolvidas e a dinâmica da cinética dos reatores. Essa abordagem multidisciplinar contribuiu para uma visão mais abrangente e aprimorada dos desafios enfrentados no campo da engenharia química e proporcionou a capacidade de aplicar soluções mais eficazes e inovadoras em projetos futuros.

## REFERÊNCIAS

- ALLOGHANI, M. et al. A Systematic Review on Supervised and Unsupervised Machine Learning Algorithms for Data Science. In: BERRY, M. W.; MOHAMED, A.; YAP, B. W. **Supervised and Unsupervised Learning for Data Science: Unsupervised and Semi-Supervised Learning**: Springer, 2020.
- ALMEIDA, M. H. B. et al. Desempenho da técnica deep learning na análise e categorização de imagens de defeito de madeira. **Energia na agricultura**, v. 33, n. 3, p. 284–291, 14 dez. 2018.
- BOCHIE, K. et al. **Aprendizado Profundo em Redes Desafiadoras: Conceitos e Aplicações**. [s.l: s.n.].
- BOSCH, R. **Indústria 4.0: muito além da automação**. 2022.
- BRAGA, A. D. P.; LUDERMIR, T. B.; CARVALHO, A. C. P. D. L. F. **Redes Neurais Artificiais: Teoria e aplicações**. Rio de Janeiro: LTC - Livros Técnicos e Científicos Editora, 2000.
- CARVALHO, R. F. **Controle preditivo baseado em modelo com estimação de estados restrita para controle e monitoramento de processos não lineares**. [s.l: s.n.].
- DAI, X.; GAO, Z. From Model, Signal to Knowledge: A **Data-Driven Perspective of Fault Detection and Diagnosis**. *IEEE Trans. Ind. Inform.* 2013, 9, 2226–2238.
- FOGLER, H. S. **Elementos de Engenharia das Reações Químicas**. 4ª. ed. Rio de Janeiro: LTC, 2009.
- FORTUNA, L. et al. **Soft Sensors for Monitoring and Control of Industrial Processes**. [s.l: s.n.].
- GAO, Z.; CECATI, C.; DING, S. X. **A survey of fault diagnosis and fault-tolerant techniques-part I: Fault diagnosis with model-based and signal-based approaches**. *IEEE Transactions on Industrial Electronics* Institute of Electrical and Electronics Engineers Inc., , 1 jun. 2015.
- GONZAGA, J. C. B. et al. ANN-based soft-sensor for real-time process monitoring and control of an industrial polymerization process. **Computers and Chemical Engineering**, v. 33, p. 43-49, 2009.
- GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [s.l: s.n.].
- KADLEC, P.; GABRYS, B.; STRANDT, S. Data-driven Soft Sensors in the process industry. **Computers and Chemical Engineering**, v. 33, n. 4, p. 795–814, 2009a.
- KINGMA, D. & BA, J. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014. LE, X. H. et al. Application of Long Short-Term Memory (*LSTM*) neural network for flood forecasting. **Water (Switzerland)**, v. 11, n. 7, 2019.

- LEE, Y. S. et al. Developing soft-sensor models using latent dynamic variational autoencoders. *IFAC-PapersOnLine*. **Elsevier B.V.**, 1 jun. 2021.
- LI, Y. et al. EA-*LSTM*: Evolutionary attention-based *LSTM* for time series prediction. **Knowledge-Based Systems**, v. 181, 1 out. 2019.
- LV, L. et al. A VMD and *LSTM* Based Hybrid Model of Load Forecasting for Power Grid Security. **IEEE Transactions on Industrial Informatics**, v. 18, n. 9, p. 6474–6482, 1 set. 2022.
- MEHDIYEV, N., LAHANN, J., EMRICH, A., ENKE, D., FETTKE, P., & LOOS, P. Time Series Classification using Deep Learning for Process Planning: A Case from the Process Industry. **Procedia Computer Science**, 114, 242–249, 2017.
- MELEIRO, L. A. D. C. **Projeto e Aplicação de Controladores Baseados em Modelos Lineares, Neurais e Nebulosos**. Tese - Universidade Estadual de Campinas. Campinas. 2002.
- NI, L. et al. Streamflow and rainfall forecasting by two long short-term memory-based models. **Journal of Hydrology**, v. 583, 1 abr. 2020.
- PARK, P. et al. Fault detection and diagnosis using combined autoencoder and long short-term memory network. **Sensors (Switzerland)**, v. 19, n. 21, 1 nov. 2019.
- RUSSELL, E. L.; CHIANG, L. H.; BRAATZ, R. D. Fault detection in industrial processes using canonical variate analysis and dynamic principal component analysis. **Chemometrics and Intelligent Laboratory Systems**, v. 51, n. 1, p. 81–93, 2000.
- SCHWAB, K. **A quarta revolução industrial**. [s.l.: s.n.].
- SILVA, G. D. C. **Estudo de um controlador preditivo não linear multivariável baseado em redes neurais**. Dissertação - Universidade Federal do Rio de Janeiro. Rio de Janeiro. 2014.
- SHAO, W.; TIAN, X. Semi-supervised selective ensemble learning based on distance to model for nonlinear soft sensor development. **Neurocomputing**, v. 222, p. 91–104, 26 jan. 2017.
- SOARES, P. L. B.; DA SILVA, J. P. Aplicação de Redes Neurais Artificiais em Conjunto com o Método Vetorial da Propagação de Feixes na Análise de um Acoplador Direcional Baseado em Fibra Ótica. **Revista Brasileira de Computação Aplicada**, v. 3, n. 2, p. 58–72, 2011.
- SOARES, F. D. R. **Técnicas de Machine Learning Aplicadas à Inferência e Detecção e Diagnóstico de Falhas de Processos Químicos Industriais em Contexto Big Data**. Dissertação (Mestrado em Ciências) - Programa de Pós-Graduação em Tecnologia de Processos Químicos e Bioquímicos, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2017.
- THIENEN, S. V. et al. Industry 4.0 and the chemicals industry: Catalyzing transformation through operations improvement and business growth. **Deloitte Iniversity Press**. 2016.
- VENKATASUBRAMANIAN, V. et al. **A review of process fault detection and diagnosis Part I: Quantitative model-based methods**. [s.l.: s.n.].



VINCENT, P.; LAROCHELL, H.; LAJOIE, I.; BENGIO, Y.; MANZAGOL, P. A. **Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion.** *J. Mach. Learn. Res.* 2010, 11, 3371–3408

VOJTĚŠEK, ING. J. **Chemical Reactors: Modern Control Methods.** 2007.

WANG, J.; LI, Y.; HAN, Z. A Novel Fault Detection Method Based on One-Dimension Convolutional Adversarial Autoencoder (1DAAE). **Processes**, v. 11, n. 2, 1 fev. 2023.

WANG, K. et al. Dynamic Soft Sensor Development Based on Convolutional Neural Networks. **Industrial and Engineering Chemistry Research**, v. 58, n. 26, p. 11521–11531, 28 maio 2019.