UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Instituto de Ciência e Tecnologia
Câmpus de Sorocaba

PHELIPE PACHLER MAIA ROMEIRO

Aplicação de Redes Neurais Recorrentes na Previsão de Demanda de Materiais: Comparação entre Modelos de Aprendizado Profundo

Sorocaba

2025

PHELIPE PACHLER MAIA ROMEIRO

APLICAÇÃO DE REDES NEURAIS RECORRENTES NA PREVISÃO DE DEMANDA DE MATERIAIS: COMPARAÇÃO ENTRE MODELOS DE APRENDIZADO PROFUNDO

Trabalho de Conclusão de Curso apresentado à Universidade Estadual Paulista (UNESP), Instituto de Ciência e Tecnologia, Sorocaba, como parte dos requisitos para obtenção do grau de Bacharel em Engenharia de Controle e Automação.

Orientador: Prof. Dr. Márcio Alexandre Marques

Sorocaba

2025

R763a

Romeiro, Phelipe Pachler Maia

Aplicação de redes neurais recorrentes na previsão de demanda de materiais : comparação entre modelos de aprendizado profundo / Phelipe Pachler Maia Romeiro. -- Sorocaba, 2025

59 p.

Trabalho de conclusão de curso (Bacharelado - Engenharia de Controle e Automação) - Universidade Estadual Paulista (UNESP), Instituto de Ciência e Tecnologia, Sorocaba

Orientador: Márcio Alexandre Marques

1. Redes neurais (Computação). 2. Aprendizado profundo (Aprendizado de máquina). 3. Inteligência artificial - Processamento de dados. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Dados fornecidos pelo autor(a).

PHELIPE PACHLER MAIA ROMEIRO

APLICAÇÃO DE REDES NEURAIS RECORRENTES NA PREVISÃO DE DEMANDA DE MATERIAIS: COMPARAÇÃO ENTRE MODELOS DE APRENDIZADO PROFUNDO

Trabalho de Conclusão de Curso apresentado ao Instituto de Ciência e Tecnologia de Sorocaba, Universidade Estadual Paulista (UNESP), como parte dos requisitos para obtenção do grau de Bacharel em Engenharia de Controle e Automação.

DATA DE DEFESA: 24/06/2025

BANCA EXAMINADORA:

Prof. Dr. Márcio Alexandre Marques

UNESP- Instituto de Ciência e Tecnologia - Campus de Sorocaba

Prof. Dr. Everson Martins

UNESP- Instituto de Ciência e Tecnologia - Campus de Sorocaba

Prof. Dr. Eduardo Verri Liberado

UNESP- Instituto de Ciência e Tecnologia - Campus de Sorocaba

AGRADECIMENTOS

Expresso minha profunda gratidão a todos que, de alguma forma, contribuíram para a realização deste trabalho e para minha jornada acadêmica.

Aos meus Orixás, pela força espiritual, proteção e intuição que me guiaram.

À minha família, em especial à minha mãe, pelo amor incondicional, apoio constante e por ser minha maior inspiração.

Ao meu orientador, Prof. Dr. Márcio Alexandre Marques, pela valiosa orientação, paciência e conhecimento transmitido, essenciais para este projeto.

Aos meus amigos, pela amizade leal, encorajamento nos desafios e pelos momentos que aliviaram a jornada.

Aos colegas de turma da Engenharia de Controle e Automação, pela parceria, troca de aprendizados e apoio mútuo durante a graduação.

A todos que me incentivaram, meu sincero obrigado.

ROMEIRO, P. P. M. **Previsão de consumo de materiais com redes neurais recorrentes para otimização da gestão de estoque industrial.** 2025. Trabalho de Conclusão de Curso (Bacharelado em Engenharia de Controle e Automação) – Instituto de Ciência e Tecnologia, Universidade Estadual Paulista, Sorocaba, 2025.

RESUMO

A otimização da gestão de estoque em ambientes fabris é crucial para a continuidade operacional e depende de previsões acuradas de consumo. Visando otimizar este processo, este trabalho investiga a aplicabilidade de redes neurais recorrentes (*Recurrent Neural Network* - RNN) na previsão de demanda de materiais. Foram desenvolvidos, treinados e comparados cinco modelos de aprendizado profundo: RNN simples, RNN com *Batch Normalization*, RNN com *Layer Normalization*, *Long Short-Term Memory* (LSTM) e *Gated Recurrent Unit* (GRU). Utilizou-se um conjunto de dados real de uma indústria de carrocerias, composto por dez séries temporais de consumo semanal com 100 amostras cada. A metodologia adotada envolveu a normalização dos dados, a criação de janelas deslizantes de 15 semanas para treino, validação e teste, e a avaliação do desempenho dos modelos com base em métricas MAE (*Mean Absolute Error*), RMSE (*Root Mean Square Error*) e MAPE (*Mean Absolute Percentage Error*), bem como, com base em previsões recursivas de 20 semanas. Os modelos foram treinados com o otimizador Adam, função de perda MSE (*Mean Square Error*) e o mecanismo de *EarlyStopping* para prevenção de sobreajuste. Os resultados demonstraram que o modelo RNN simples obteve o melhor desempenho, com um MAPE de 1,785%, superando as arquiteturas mais complexas LSTM, GRU e as variantes de RNN com normalização. Concluiu-se que, para o volume e as características dos dados analisados, a menor complexidade da RNN Simples proporcionou uma melhor capacidade de generalização. Além disso, apontou-se que a utilização de séries temporais mais longas ou a inclusão de variáveis exógenas pode contribuir para o aprimoramento das previsões futuras.

Palavras-chave: gestão de estoque; previsão de demanda de materiais; redes neurais recorrentes; séries temporais.

ROMEIRO, P. P. M. **Material consumption forecasting with recurrent neural networks for optimizing industrial inventory management.** 2025. Final paper (Bachelor's degree in Control and Automation Engineering) – Institute of Science and Technology, São Paulo State University, Sorocaba, 2025.

ABSTRACT

Optimizing inventory management in manufacturing environments is crucial for operational continuity and relies on accurate consumption forecasting. Aiming to enhance this process, this study investigates the applicability of Recurrent Neural Networks (RNNs) in material demand forecasting. Five deep learning models were developed, trained, and compared: Simple RNN, RNN with Batch Normalization, RNN with Layer Normalization, Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU). A real-world dataset from a bus body manufacturing company was used, consisting of ten weekly consumption time series with 100 samples each. The adopted methodology included data normalization, the creation of 15-week sliding windows for training, validation, and testing, and performance evaluation based on MAE (Mean Absolute Error), RMSE (Root Mean Square Error), and MAPE (Mean Absolute Percentage Error) metrics, using 20-week recursive forecasts. The models were trained using the Adam optimizer, Mean Square Error (MSE) loss function, and the EarlyStopping mechanism to prevent overfitting. Results demonstrated that the simple RNN model achieved the best performance, with a MAPE of 1.785%, outperforming the more complex LSTM, GRU, and the RNN variants with normalization. It was concluded that, for the volume and characteristics of the analyzed data, the lower complexity of the simple RNN provided superior generalization capacity. Furthermore, it was suggested that the use of longer time series or the inclusion of exogenous variables may contribute to the enhancement of future forecasts.

Keywords: inventory management; material demand forecasting; recurrent neural networks; time series.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

RNN – *Recurrent Neural Network*

LSTM – *Long Short-Term Memory*

GRU – *Gated Recurrent Unit*

MAE – *Mean Absolute Error*

RMSE – *Root Mean Square Error*

MAPE – *Mean Absolute Percentage Error*

MSE – *Mean Square Error*

PPCP – Planejamento, programação e controle de produção

PE – Ponto de encomenda

AM – Aprendizado de máquina

ML – *Machine learning*

AP – Aprendizado profundo

DL – *Deep learning*

RNA – Rede Neural Artificial

BN – *Batch Normalization*

LN – *Layer Normalization*

CNN – *Convolutional Neural Network*

SUMÁRIO

1 INTRODUÇÃO

De acordo com Lustosa *et al.* (2008), a produção de bens de consumo duráveis e não duráveis, da maneira como se dá atualmente, apenas teve início com a Revolução Industrial, período em que se tornou possível produzir e criar meios para o consumo em massa. Neste contexto, os sistemas de Planejamento, Programação e Controle da Produção - PPCP se desenvolveram como resultado da evolução da ciência da administração, tratando-a como ciência baseada na observação, medição, análise e aprimoramento dos métodos de trabalho.

O PPCP é uma função da administração que integra a produção às demais atividades de uma empresa, por meio da informação. Pela definição, planejar é projetar o futuro de maneira diferente do passado, baseado no controle obtido dos processos produtivos. Por sua vez, controlar é manejar variações e possíveis desvios que acarretam a necessidade de redesenhar planos ou intervir em operações (Graziani, 2012).

Inserido à uma empresa, o PPCP é uma área de apoio responsável pela coordenação e aplicação dos recursos produtivos, da melhor forma possível, aos planejamentos estabelecidos nos níveis estratégico, tático e operacional, como afirma Tubino (2007). De acordo com Martins *et al.* (2007), o PPCP deve possuir e informar dados acerca da situação corrente dos recursos, compreendendo pessoas, equipamentos, instalações, materiais e ordens de fabricação e de compra, além de ser capaz de realizar tomadas de decisões eficazes. As informações devem estar disponíveis e atualizadas, para que se possa oferecer aos clientes uma ampla variedade de serviços, melhorar o planejamento, a programação e o controle em um ambiente de negócios internacionalizado; e que a habilidade da empresa nestes aspectos poderá ser o diferencial para que ela seja de classe mundial, acrescentando também que a informação deve estar disponível no chão de fábrica. Além disso, a competência da empresa nessas áreas pode ser crucial para alcançar padrões de classe mundial.

Assim, o PPCP constitui um sistema de informações intimamente ligado à estratégia de manufatura, fornecendo suporte às decisões táticas e operacionais. Ele aborda questões essenciais sobre o que produzir e comprar, em que quantidade, quando e com quais recursos. Na prática, a programação da produção busca garantir uma elevada taxa de utilização das instalações, enquanto a sequência da programação dos produtos visa minimizar os tempos de *setup* (período de interrupção da produção para ajustes dos equipamentos fabris) (Martins *et al.*, 2007).

Segundo Totvs (2020), a estrutura de um sistema de PPCP pode ser definida pelo planejamento estratégico da produção, com objetivo de minimizar riscos nas tomadas de decisão, estabelecendo um plano de produção a longo prazo, a partir da previsão das vendas e da disponibilidade de recursos; pelo planejamento mestre da produção, que consiste em uma programação de curto prazo para produtos finais; pela programação de produção, que estabelece quantidades e prazos para dimensionamento das ordens de compra e fabricação; e pelo monitoramento e controle da produção, garantindo que todo o programa seja realizado conforme o planejado, através da análise de dados e aplicação de medidas corretivas, de forma ágil (Totvs, 2020).

Nesse contexto, a previsão de demanda se apresenta como a variável mais importante para as funções desempenhadas pela área de PPCP, sendo a base para planejamentos estratégicos da produção, de vendas e de finanças de uma empresa. Através de uma eficaz previsão de demanda, empresas podem desenvolver planos de capacidade, fluxo de caixa, vendas, produção, estoque, mão-de-obra, compras, entre outros. As previsões permitem visualizar e analisar cenários futuros e planejar adequadamente planos de ações (Tubino, 2007).

Com base na previsão de demanda para um determinado período, é possível estimar as quantidades de materiais necessárias para atender às vendas previstas — ou seja, o consumo previsto. Segundo Totvs (2018), a partir desse consumo estimado, do estoque de segurança (quantidade fixa estabelecida para cobrir variações inesperadas na demanda ou atrasos no fornecimento) e do tempo total de ressuprimento (composto pelo tempo interno do processo de compras e pelo prazo de entrega do fornecedor), calcula-se o ponto de encomenda (PE).

O PE representa o nível mínimo de estoque a partir do qual deve ser iniciada automaticamente uma nova compra ou produção, garantindo que os materiais sejam repostos a tempo. Esse método de controle de estoque, conhecido como ressuprimento por ponto de encomenda, tem como principal objetivo evitar rupturas no abastecimento durante o período de reposição, além de contribuir para a redução de falhas operacionais e melhor eficiência no gerenciamento dos estoques (Totvs, 2018).

É possível realizar a previsão de demanda através de modelos de previsão quantitativa. Conforme Amazon (2023a), estes modelos usam informações estatísticas relevantes e dados históricos para prever tendências futuras. Como exemplos temos a modelagem econométrica, que utiliza dados financeiros para prever mudanças econômicas significativas e seus impactos na empresa e a previsão de séries temporais, que analisa dados sequenciados ao longo do tempo

para prever tendências futuras destes dados. A previsão de séries temporais pode ser realizada por métodos estatísticos, de aprendizado de máquina e aprendizado profundo (Amazon, 2023a).

De acordo com IBM (2015) o aprendizado de máquina (AM), conhecido como *machine learning* (ML), é um subcampo da engenharia e da ciência da computação, o qual utiliza computadores com a capacidade de aprender através de associações de diferentes dados, padrões e comportamentos em diversas áreas de atuação.

Um algoritmo de *machine learning* utiliza um conjunto de dados denominado como dados de treino, para aprender os padrões existentes neste grupo, e assim, criar funções para gerar respostas. As respostas ou resultados do algoritmo são comparados com os dados esperados para este conjunto, denominados como dados de teste, para avaliação de seu desempenho (IBM, 2015).

Já o aprendizado profundo (AP) (*deep learning* (DL)), é um subconjunto do *machine learning* que surgiu da tentativa de aumentar a eficácia das técnicas tradicionais de ML. De acordo com Amazon (2023b), as técnicas tradicionais de ML costumam exigir considerável esforço humano para treinar o modelo, característica essa conhecida como aprendizado supervisionado. Dessa forma, o DL apresenta benefícios em relação ao ML tradicional, devido a sua capacidade de processar eficientemente dados não estruturados, encontrar relacionamentos ocultos entre eles e descobrir padrões, realizar aprendizado sem supervisão e processar dados voláteis, ou seja, dados que podem apresentar grandes variações, como por exemplo, dados bancários. Os modelos de DP conseguem realizar previsões precisas, encontrar padrões complexos em dados de imagens, textos, sons e outros, ou até automatizar tarefas, como descrever imagens ou transcrever arquivos de som para texto escrito (Amazon, 2023b).

De acordo com Academy (2023), no âmbito da modelagem de séries temporais, os algoritmos de DL, como redes neurais recorrentes, *long short-term memory* (LSTM) e *gated recurrent unit* (GRU) provam ser particularmente eficazes no reconhecimento de padrões em séries temporais. A principal vantagem que estes algoritmos apresentam em relação as clássicas técnicas estatísticas, é a sua capacidade de modelar interações e comportamentos não lineares complexos.

Dado o contexto de planejamento e controle de produção, de gestão e previsão de demanda e dos métodos de previsão de séries temporais, o presente trabalho visa implementar modelos de previsão utilizando algoritmos de DL baseados em RNN, LSTM e GRU, com intuito de prever a demanda de materiais de uma empresa (consumo previsto), permitindo assim a aplicação do método de ressuprimento por ponto de encomenda, importante para uma gestão de estoque eficaz. O projeto propõe também a análise dos resultados obtidos pelos modelos, bem como possíveis tratamentos e ajustes que possam promover melhores desempenhos dos modelos construídos.

2 REVISÃO BIBLIOGRÁFICA

Esta seção tem como objetivo apresentar os fundamentos teóricos essenciais que embasam o desenvolvimento deste trabalho. A revisão abrange desde os conceitos fundamentais de séries temporais e técnicas de análise exploratória de dados, passando pelo estudo aprofundado das redes neurais recorrente e suas arquiteturas proeminentes para previsão, até as práticas de treinamento, avaliação de modelos e as ferramentas computacionais, como por exemplo, as bibliotecas Python, cruciais para a implementação da solução proposta.

2.1 Séries temporais

Conforme descreve Nielsen (2021), as séries temporais são conjuntos de dados organizados e coletados ao longo do tempo, geralmente em intervalos regulares (como dias, meses ou anos). Esses dados representam a evolução de uma ou mais variáveis durante o período. São amplamente utilizadas em várias áreas, como economia, finanças, meteorologia, saúde e ciência de dados, com objetivo de analisar tendências, padrões sazonais, flutuações e prever valores futuros.

Os dados e análises de séries temporais assumem uma importância cada vez maior, em função da produção volumosa desses mesmos dados devido à Internet das Coisas, à digitalização dos sistemas de assistência médica e às cidades inteligentes, conforme Nielsen (2021). Esses fatores, atrelados à competitividade de mercado, torna a relevância dos dados de séries temporais cada vez maior.

Assim, cada vez mais técnicas robustas e eficientes de análise de séries temporais, principalmente apoiados por estatística e aprendizado de máquina, são implementadas. Essas técnicas são úteis para predição de comportamento humano, fenômenos científicos e dados do setor privado, refletindo em campos de atuação oportunos para produção de uma rica variedade de dados de séries temporais (Nielsen, 2021).

A análise de séries temporais geralmente se resume a causalidade, buscando compreender como o passado influencia o futuro, se apresentando como relevantes em diversos campos de atuação. Dessa maneira, várias especialidades contribuíram com diferentes formas de observar e analisar as séries temporais (Nielsen, 2021).

2.2 Fundamentos da análise exploratória de dados

A análise exploratória de dados consiste em cálculos de medidas estatísticas que resumem as informações obtidas, dando uma visão global dos dados. Conforme descreve Medri (2011), sua importância reside justamente em prover essa visão global e sintetizada do conjunto de dados por meio de medidas conhecidas como medidas descritivas. Quando calculadas com dados amostrais, elas são chamadas de estatísticas, e de parâmetros quando se referem a dados populacionais.

As principais medidas descritivas utilizadas no contexto de dados de séries temporais são medidas de tendência central (como média aritmética e mediana), medidas de dispersão (como desvio padrão e coeficiente de variação) e medidas de formato da distribuição (como assimetria e curtose). De acordo com Piana *et al.* (2013), medidas de tendência central tem o objetivo de representar o ponto de equilíbrio, centro ou valor típico de uma distribuição de dados, sintetizando informações dessa distribuição. Medidas de dispersão produzem informações sobre a variabilidade dos dados, indicando quanto as observações diferem entre si ou o grau de afastamento das observações em relação à média. Medidas de formato informam sobre o modo em que os valores se distribuem, indicando por exemplo se as maiores concentrações de valores estão no centro ou nas extremidades, ou o grau de achatamento da curva de distribuição.

Dentre as medidas de tendência central, a média aritmética (equação (1)) representa um valor central da distribuição, mas que pode ser afetada por valores extremos. Dessa maneira, seu uso é recomendado em distribuições simétricas ou levemente assimétricas, e em distribuições não heterogêneas, pois em outros casos pode perder a sua capacidade de representar a distribuição. Em contrapartida, a mediana (equação (2)) é definida pelo valor que ocupa a posição central de um conjunto de valores ordenados, dividindo a distribuição em duas partes iguais. A vantagem da mediana perante a média é a de não ser tão afetada pelos valores extremos, tornando-se uma medida robusta e mais resistente a observações discrepantes (Piana *et al.*, 2013).

$$\bar{X} = \frac{1}{n}\sum_{n=1}^{n} x_i \tag{1}$$

Onde:

- $\bar{X}$: média aritmética
- n: quantidade de valores
- x_i: valores do conjunto de dados

$$\tilde{X} = \begin{cases} X\left[\frac{n+1}{2}\right] & se\ n\ for\ ímpar \\ \frac{X\left[\frac{n}{2}\right] + X\left[\frac{n}{2}+1\right]}{2} & se\ n\ for\ par \end{cases} \quad (2)$$

Onde:

- $\tilde{X}$: mediana
- X: lista ordenada de valores do conjunto de dados

De acordo com Medri (2011), dentre as medidas de dispersão, o desvio padrão (equação (3)) é calculado pela raiz quadrada da variância. A variância (equação (4)), por sua vez, é calculada elevando-se os desvios de cada valor em relação à média ao quadrado, potencializando os afastamentos e enfatizando os grandes desvios. A medida quantifica o grau de agregação dos dados em torno da média. Já o coeficiente de variação (equação (5)) é uma medida adimensional, frequentemente expressa em percentual, utilizada para comparar o grau de concentração em torno da média. Uma distribuição é considerada homogênea quando o coeficiente de variação for menor que 20%.

$$\sigma = \sqrt{\frac{\sum(x_i + \bar{X})^2}{n}} \quad (3)$$

Onde:

- σ: desvio padrão

$$S^2 = \frac{\sum(x_i + \bar{X})^2}{n} \quad (4)$$

Onde:

- S^2: variância

$$CV = \frac{\sigma}{\bar{X}} \tag{5}$$

Onde:

- CV: coeficiente de variação

Conforme Medri (2011), a assimetria e a curtose referem-se à forma da distribuição dos dados. A assimetria (equação (6)) indica se a distribuição é simétrica ou se apresenta uma cauda em uma das extremidades, sendo positiva se a cauda está à direita, e negativa caso seja à esquerda. Ela permite verificar a medida de concentração dos dados perante os valores de tendência central, e é considerada moderada se estiver entre 0,15 e 1,00, e forte se for maior que 1,00. Por sua vez, a curtose (equação (7)) é o grau de achatamento de uma distribuição em relação a uma distribuição padrão (curva normal). A curval normal é a referência da curtose, sendo denominada de mesocúrtica. Se a distribuição apresentar uma curva de frequência mais achatada do que a normal, é denominada de leptocúrtica, e se apresentar frequência mais aberta, é chamada de platicúrtica. Distribuições com curtose alta tendem a ter caudas maiores, indicando maior probabilidade de valores discrepantes em relação a uma distribuição normal.

$$\tilde{\mu}_3 = \frac{\sum_{i=1}^{n}(x_i - \bar{X})^3}{(n-1) \times \sigma^3} \tag{6}$$

Onde:

- $\tilde{\mu}_3$: assimetria

$$K = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{X})^4}{\sigma^4} \tag{7}$$

Onde:

- K: curtose

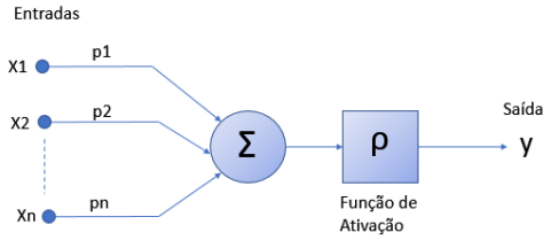
Dessa maneira, de acordo com Hyndman *et al*. (2021), a análise exploratória de séries temporais permite caracterizar profundamente a série, compreendendo seu nível típico, variabilidade e natureza das flutuações, além de auxiliar na decisão sobre transformações necessárias para normalizar a distribuição e melhorar a precisão de previsões.

2.3 Redes neurais recorrentes para previsão de séries temporais

2.3.1 Arquiteturas de redes neurais recorrentes

As redes neurais recorrentes se baseiam na estrutura das redes neurais artificiais (RNA). Conforme Souza (2018), o conhecimento das redes neurais é adquirido por meio de seu processo de aprendizagem e das conexões entre os "neurônios", definidos como pesos sinápticos, que armazenam o conhecimento obtido. O algoritmo de aprendizagem das redes neurais é essencialmente um processo no qual os pesos sinápticos são ajustados sistematicamente durante o treinamento, permitindo que a rede seja capaz de reconhecer padrões de acordo com a sua concepção. A Figura 1 apresenta um modelo matemático do "neurônio". Cada entrada $\mathbf{X_i}$ do "neurônio" é modelada a partir dos pesos sinápticos $\mathbf{P_n}$, que são fatores pré-determinados pelo "neurônio". Sua saída final (**Y**) é resultado da aplicação de uma função de ativação sobre a saída intermediária da célula.

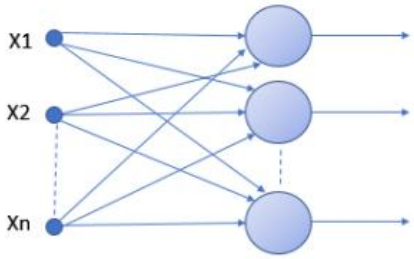
Figura 1 - Modelo matemático do "neurônio"



Fonte: (Souza, 2018).

De acordo com Souza (2018), uma rede neural possui camadas definidas por diferentes formas de agrupamento dos "neurônios". Redes neurais de camada única possuem apenas uma camada e um nó entre a entrada e a saída da rede, como mostra a Figura 2. Redes com mais de uma camada de "neurônios" entre as camadas de entrada e de saída, são denominadas redes multicamadas, cujas camadas intermediárias são chamadas de camadas ocultas, e seus sinais passam para os próximos neurônios através de funções de transferência presentes em cada "neurônio", como mostra a Figura 3.

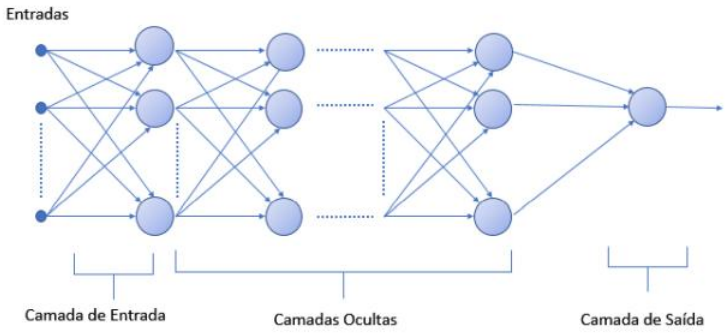
Figura 2 - Rede neural de camada única



Fonte: (Souza, 2018).

As redes neurais podem ser totalmente conectadas ou parcialmente conectadas. Em uma rede totalmente conectada, cada "neurônio" possui ligação com todos os outros "neurônios" da camada subsequente, enquanto redes parcialmente conectadas os "neurônios" não se conectam a todos da próxima camada (Souza, 2018).

Figura 3 - Rede neural de multicamada



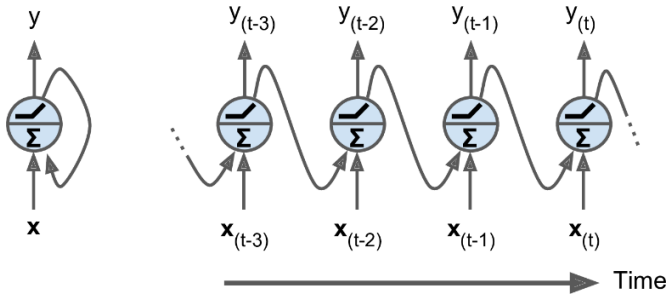
Fonte: (Souza, 2018).

Segundo Souza (2018), as RNA são classificadas de acordo com a forma como os sinais se propagam na rede. Redes neurais progressivas ou redes *feedforward*, são totalmente conectadas. Dessa maneira, o fluxo dos sinais segue apenas um sentido, da camada de entrada até a camada de saída. De acordo com Géron (2021), as redes neurais recorrentes são totalmente conectadas, mas também possuem conexões *backward*, ou seja, o sinal pode retornar para a camada anterior.

Na Figura 4 à esquerda, vemos uma RNN da forma mais simples possível, composta de um "neurônio" que recebe entradas, gera uma saída e envia esse sinal de volta para si mesmo. A cada intervalo de tempo **t**, denominado também de *frame*, este "neurônio" recebe as entradas $\mathbf{x}_{(t)}$, além de seu próprio intervalo de tempo anterior $\mathbf{y}_{(t-1)}$. O primeiro frame, geralmente $\mathbf{y}_{(t-1)}$, é definido como zero. À direita da Figura 4 temos a representação do comportamento desse "neurônio" ao longo do tempo, chamado de *unrolling the network through time* (desenrolando

a rede ao longo do tempo), sendo o mesmo “neurônio” recorrente representado através do tempo (Géron, 2021).

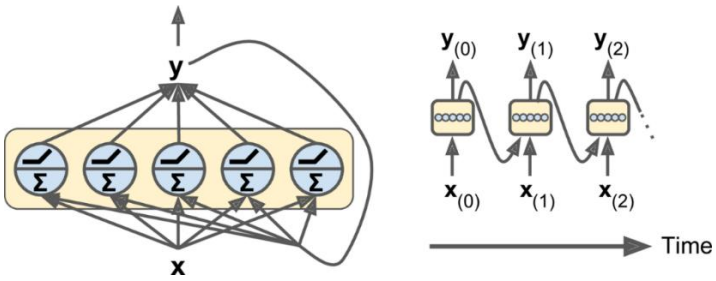
Figura 4 - Um neurônio recorrente (à esquerda) se desenrolando ao longo do tempo (à direita)



Fonte: (Géron, 2021).

Uma camada de “neurônios” recorrentes é representada na Figura 5, à esquerda. A cada intervalo de tempo **t**, cada neurônio recebe o vetor de entrada $\mathbf{x_{(t)}}$ e o vetor de saída do intervalo de tempo anterior $\mathbf{y_{(t-1)}}$, da mesma maneira que a rede neural composta de apenas um neurônio, com a diferença de que as saídas passam a ser vetores de sinais, ao invés de um sinal escalar. A Figura 5, à direita, mostra a camada de “neurônios” recorrentes desenrolada ao longo do tempo (Géron, 2021).

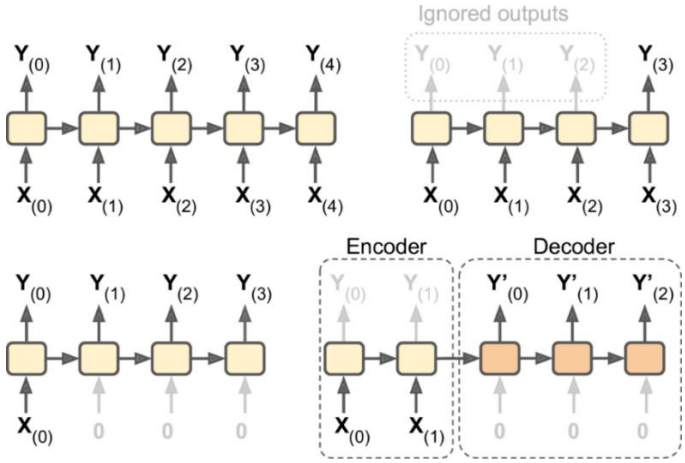
Figura 5 - Uma camada de neurônios recorrentes (à esquerda) desenrolada ao longo do tempo (à direita)



Fonte: (Géron, 2021).

Uma RNN pode receber uma sequência de entradas simultâneas e gerar uma sequência de saídas, como mostra a rede superior esquerda da Figura 6. Redes deste tipo são chamadas de redes *sequence-to-sequence* (redes sequência a sequência) e são úteis para a predição de séries temporais, como por exemplo, preços de ações, onde é possível fornecer os preços dos últimos **N** dias, e a rede neural pode gerar como saída a variação dos preços em um dia futuro, a partir de **N-1** dias até o próximo dia (Géron, 2021).

Figura 6 - Redes sequência a sequência (canto superior esquerdo), sequência a vetor (canto superior direito), vetor à sequência (canto inferior esquerdo) e redes codificador-decodificador (canto inferior direito)



Fonte: (Géron, 2021).

Outra maneira de usar essa estrutura de rede é fornecer uma sequência de entrada e ignorar as saídas, exceto pela última, como mostra a rede na parte superior direita da Figura 6. Como exemplo de uso, é possível fornecer uma sequência de palavras, como a resenha de um filme, cuja resposta da rede seria uma pontuação da opinião dada na resenha, sendo de **-1** (opinião bastante negativa) a **+1** (opinião bastante positiva) (Géron, 2021).

É possível também fornecer à rede o mesmo vetor de entrada, repetidamente a cada intervalo de tempo, e deixá-la gerar uma sequência de saídas, chamada de *vector-to-sequence-network* (redes vetor à sequência). A imagem da parte inferior esquerda da Figura 6 demonstra uma rede desse tipo. Tal configuração pode ser utilizada com uma imagem sendo fornecida na entrada e a saída poderia ser uma legenda para essa imagem (Géron, 2021).

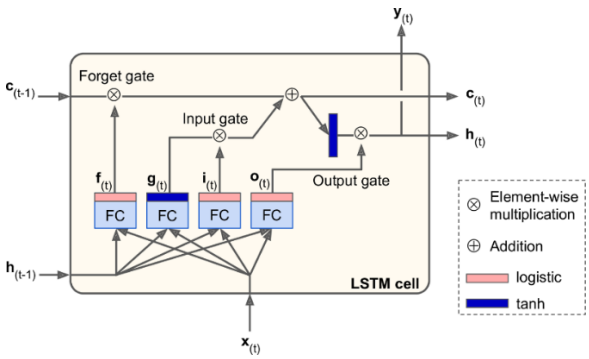
Por fim, é possível ter uma rede de sequência a vetor, chamada codificador, acompanhada de uma rede vetor à sequência, chamada decodificador, como mostra a imagem inferior da direita da Figura 6. Essa estrutura pode ser usada em traduções de textos, por exemplo, fornecendo uma frase à rede, onde o codificador converteria a frase em uma representação vetorial, e o decodificador decodificaria esse vetor em uma frase em outro idioma. Esse modelo é denominado *encoder-decoder* (codificador-decodificador) e funciona melhor em traduções em comparação com redes sequência a sequência, pois as palavras finais da frase podem influenciar na tradução das primeiras palavras. Dessa maneira, é mais interessante que a RNN receba a frase inteira antes de realizar a tradução (Géron, 2021).

2.3.2 Arquitetura LSTM

Conforme (Géron, 2021), a célula LSTM (*long short-term memory*) é um tipo especial de RNN, projetada para aprender dependências de longo prazo em dados sequenciais. Ela consegue isso através de "portões" que controlam o fluxo da informação, permitindo que a rede retenha ou esqueça informações seletivamente ao longo do tempo.

Uma célula LSTM possui seu estado dividido em dois vetores, $\mathbf{h_{(t)}}$ e $\mathbf{c_{(t)}}$, onde $\mathbf{h_{(t)}}$ representa o estado de curto prazo e $\mathbf{c_{(t)}}$ o estado de longo prazo. A Figura 7 mostra a arquitetura de uma célula LSTM (Géron, 2021).

Figura 7 - Célula LSTM



Fonte: (Géron, 2021).

Como o estado de longo prazo $\mathbf{c_{(t-1)}}$ atravessa a rede da esquerda para a direita, ele passa por um "portão" de esquecimento, esquecendo algumas memórias e, em seguida, adiciona novas memórias por meio da operação de adição, que adiciona as memórias que foram selecionadas por um "portão" de entrada. Após isso, o resultado $\mathbf{c_{(t)}}$ é enviado diretamente para a saída da célula, sem qualquer transformação adicional. Assim, a cada intervalo de tempo, algumas memórias são esquecidas e outras são adicionadas. Além disso, após a adição, o estado de longo prazo é copiado e passado pela função **tanh** e, em seguida, o resultado é filtrado pelo "portão" de saída. Esse processo gera o estado de curto prazo $\mathbf{h_{(t)}}$, igual à saída da célula para o intervalo de tempo $\mathbf{y_{(t)}}$ (Géron, 2021).

O vetor de entrada atual $\mathbf{x_{(t)}}$ e o estado anterior de curto prazo $\mathbf{h_{(t-1)}}$ são fornecidos a quatro camadas totalmente conectadas e que possuem propósitos diferentes. A camada principal gera a saída $\mathbf{g_{(t)}}$ que normalmente analisa as entradas atuais $\mathbf{x_{(t)}}$ e os estados anteriores de curto prazo $\mathbf{h_{(t-1)}}$. Diferente de células simples, em uma célula LSTM as partes mais importantes dessa saída são armazenadas no estado de longo prazo. As outras três camadas são controladores de "portões" (*gate controllers*) e uma vez que usam a função de ativação logística, suas saídas variam de **0** a **1**. Elas são alimentadas e, se elas gerarem **1** como saída, abrem o "portão". Assim,

o portão de esquecimento, controlado por $\mathbf{f}_{(t)}$, controla quais partes do estado de longo prazo devem ser esquecidas; o "portão" de entrada, controlado por $\mathbf{i}_{(t)}$, controla quais partes da saída $\mathbf{g}_{(t)}$ devem ser adicionadas ao estado de longo prazo; e o "portão" de saída, controlado por $\mathbf{o}_{(t)}$, controla quais partes do estado de longo prazo devem ser lidas e geradas nesse intervalo de tempo, tanto para o estado de curto prazo quanto para a saída da célula (Géron, 2021).

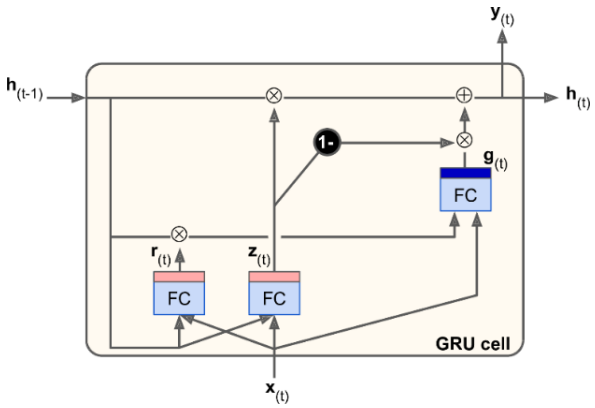
Em suma, uma célula LSTM pode aprender a reconhecer uma entrada importante (função do "portão" de entrada), armazená-la no estado de longo prazo e preservá-la pelo tempo que for necessário (função do "portão" de esquecimento), além de extraí-la sempre que necessário. Esse fato explica o motivo dessas células terem sido tão bem-sucedidas na identificação de padrões de longo prazo em séries temporais (Géron, 2021).

2.3.3 Arquitetura GRU

A célula GRU (*gated recurrent unit*) é uma versão simplificada da célula LSTM, mas sem perder sua eficácia de funcionamento. De acordo com Géron (2021), dentre suas facilitações estão que ambos os vetores de estados são incorporados em um único vetor $\mathbf{h}_{(t)}$ e um único controlador de "portão" $\mathbf{z}_{(t)}$ que controla os portões de esquecimento e de saída. Se $\mathbf{z}_{(t)}$ for igual a **1**, o "portão" de esquecimento abre e o "portão" de saída fecha. Se a saída for **0**, acontece o oposto. A Figura 8 apresenta a arquitetura de uma célula GRU.

Em suma, sempre que uma memória tem que ser armazenada, esse local é o primeiro a ser esquecido. Outra facilitação é que não há "portão" de saída, assim, o vetor de estado completo é gerado a cada intervalo de tempo. Porém, há um novo controlador de portão $\mathbf{r}_{(t)}$ que controla qual parte do estado anterior será exibida para a camada principal ($\mathbf{g}_{(t)}$) (Géron, 2021).

Figura 8 - Célula GRU



Fonte: (Géron, 2021).

2.3.4 Técnica de *Batch Normalization*

A *Batch Normalization* (BN) é uma técnica usada para melhorar a velocidade de treinamento de um modelo, o desempenho e a estabilidade de redes neurais. Conforme Géron (2021), a BN tem como objetivo centralizar o zero e normalizar cada entrada da camada, subtraindo a média do lote e dividindo pelo desvio padrão do lote. Em seguida, ela reescala e desloca esses valores normalizados, permitindo que o modelo aprenda o escalonamento e a média ideais de cada uma das entradas da camada (ativações normalizadas). Isso reduz o problema da mudança na distribuição das entradas das camadas internas da rede durante o treinamento (atualização dos pesos ao longo das camadas), permitindo taxas de aprendizado mais altas e tornando a rede menos sensível a inicialização de pesos.

2.3.5 Técnica de *Layer Normalization*

A *Layer Normalization* (LN) é outra técnica de normalização semelhante à normalização em *batch*, mas ao invés de normalizar na dimensão do lote, ela normaliza na dimensão das características. Dessa maneira, a normalização ocorre a partir de todas as ativações somadas dentro de uma única camada para uma única amostra de treinamento (Géron, 2021).

De acordo com Géron (2021), a LN possui a vantagem de calcular as estatísticas necessárias (média e desvio padrão) instantaneamente em cada intervalo de tempo e de forma independente para cada amostra. Isso também significa que ela se comporta da mesma maneira durante o treinamento e o teste, diferente da normalização em *batch*. Ela não precisa usar as médias móveis exponenciais para estimar as estatísticas das características em todas as amostras no conjunto de treinamento.

2.4 Treinamento e avaliação de modelos

2.4.1 Separação de dados de treino, validação e teste

Conforme Suniga (2020) descreve, a divisão dos dados em conjuntos de treino, validação e teste é uma prática fundamental em ML, visando garantir a generalização, que é a capacidade do modelo de aplicar seu aprendizado eficazmente a dados não utilizados durante o treinamento. O conjunto de dados de treinamento são usados para treinar o modelo; os dados de validação servem para comparar diferentes modelos e seus hiperparâmetros, além de permitir a realização

de múltiplos experimentos; e os dados de teste são empregados para comprovar que o modelo final realmente funciona e generaliza bem.

Para exemplificação de separação utilizando dados temporais, uma seleção puramente aleatória dos conjuntos poderia não representar adequadamente um cenário de previsão futura. Para um histórico de seis meses de dados, a abordagem sugerida é utilizar os dados do primeiro ao quarto mês para treino. Os dados do quinto mês seriam designados para validação, e os do sexto mês para teste. Em casos de dados temporais, os conjuntos de validação e teste devem idealmente conter os dados mais recentes, e o conjunto de validação deve ser representativo do conjunto de teste, a fim de evitar a criação de modelos enviesados por eventos raros não representativos do comportamento futuro (Suniga, 2020).

2.4.2 Curvas de aprendizado, subajuste e sobreajuste

As curvas de aprendizado são gráficos que representam o desempenho de um modelo de ML, especialmente aqueles que aprendem incrementalmente ao longo do tempo, como redes neurais de aprendizado profundo, em função da experiência ou tempo de treinamento. De acordo com Brownlee (2019), estas curvas utilizam métricas que podem ser de minimização (como perda ou erro, onde valores menores são melhores e 0,0 indica aprendizado perfeito do treino) ou de maximização (como acurácia, onde valores maiores são melhores). Essas curvas se distinguem entre curvas de aprendizado de otimização (baseadas na métrica pela qual os parâmetros do modelo são otimizados, como a métrica de perda) e curvas de aprendizado de desempenho (baseadas na métrica pela qual o modelo será avaliado e selecionado, como a acurácia).

Como afirma Brownlee (2019), as curvas de aprendizado são ferramentas de diagnóstico cruciais, tipicamente facilitando a análise do desempenho no conjunto de treinamento (indicando quão bem o modelo está aprendendo) e outra para o desempenho em um conjunto de validação separado, que não faz parte do treinamento (indicando quão bem o modelo está generalizando). A análise da forma e da dinâmica dessas curvas permite diagnosticar diferentes comportamentos e problemas, como o subajuste (*underfitting*), sobreajuste (*overfitting*) e o bom ajuste (*good fit*).

O subajuste é diagnosticado quando o modelo não consegue aprender o conjunto de treinamento adequadamente. A curva de perda de treinamento pode se apresentar plana ou com valores ruidosos e relativamente altos, indicando incapacidade de aprender. Também pode

continuar diminuindo até o final do gráfico, sugerindo que o treinamento foi interrompido prematuramente e o modelo ainda era capaz de melhorias. Isso pode ocorrer quando o modelo não tem capacidade adequada para a complexidade do conjunto de dados (Brownlee, 2019).

O sobreajuste é caracterizado quando o modelo aprendeu o conjunto de treinamento excessivamente bem, incluindo ruído estatístico ou flutuações aleatórias. A curva de perda de treinamento continua a diminuir com a experiência, mas a curva de perda de validação suaviza até um ponto de inflexão e então começa a aumentar novamente. Este comportamento indesejado prejudica a generalização para novos dados, frequentemente ocorre se o modelo tem mais capacidade do que o necessário ou é treinado por tempo demais (Brownlee, 2019).

O bom ajuste é o cenário ideal, onde as perdas de treinamento e validação diminuem e convergem para um ponto de estabilidade, mantendo uma pequena e aceitável diferença (a lacuna de generalização) entre os valores finais de perda. Treinamento contínuo de um bom ajuste provavelmente levará a um sobreajuste (Brownlee, 2019).

Adicionalmente, as curvas de aprendizado podem ajudar a identificar se os conjuntos de dados não são relativamente representativos um do outro ou do domínio do problema, o que pode ocorrer se um dos conjuntos tiver um número de amostras muito menor em relação ao outro. Um conjunto de treinamento não representativo pode ser indicado por uma curva de perda de treinamento que melhora, assim como a de validação, mas uma grande lacuna permanece entre ambas as curvas (Brownlee, 2019).

De acordo com Brownlee (2019), um conjunto de validação não representativo (por exemplo, com poucas amostras) pode apresentar uma curva de validação com movimentos ruidosos em torno da curva de treinamento. Se a perda de validação for consistentemente menor que a perda de treinamento, isso pode indicar que o conjunto de validação é, na verdade, mais fácil para o modelo prever do que o próprio conjunto de treinamento.

2.4.3 Métricas de erro

Segundo Penteado (2021), as medidas de erro são calculadas a partir dos valores de previsão obtidos, a fim de verificar a acuracidade do modelo. As mais comumente utilizadas são o erro médio absoluto (MAE), a raiz quadrada do erro-médio (RMSE) e a média percentual absoluta do erro (MAPE).

O erro médio absoluto é calculado através da equação 8, onde **n** é o número de amostras da série, **Y** é o valor real e $\hat{\mathbf{Y}}$ é o valor previsto daquele mesmo ponto da série temporal. Este erro possui a vantagem de sua facilidade de ser calculado, porém seu método oculta a presença de valores que fujam da normalidade (*outliers*) que podem comprometer o processo de previsão (Penteado, 2021).

$$MAE = \frac{\sum_{i=1}^{n} |Y_i - \hat{Y}_i|}{n} \tag{8}$$

A raiz quadrada do erro-médio é calculada através da equação 9. Sua grande diferença em comparação ao MAE é a presença do termo quadrático entre os valores reais e previstos, e a raiz quadrada do valor final. Dessa forma, por elevar os termos à sua segunda potência, a RMSE se torna mais sensível a *outliers* em relação ao MAE, podendo ser mais interessante na identificação destes dados diferenciados (Penteado, 2021).

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (Y_i - Y_i)^2}{n}} \tag{9}$$

Já a média percentual absoluta do erro (MAPE), apresentada na equação 10, é uma métrica que expressa o erro médio em termos percentuais. Para o seu cálculo, o erro médio absoluto (MAE) é dividido pelo valor real correspondente (**Y**) e multiplicado por 100%, para obtenção da representação percentual. Por ser uma métrica percentual, a MAPE é bastante interessante para análises e comparações de erros percentuais de modelos de previsão (Penteado, 2021).

$$MAPE = \frac{1}{n} \frac{\sum_{i=1}^{n} |Y_i - \hat{Y}_i|}{Y_i} \times 100\% \tag{10}$$

2.5 Bibliotecas Python

No desenvolvimento deste projeto, o uso de bibliotecas Python é fundamental. Elas tornam o trabalho muito mais prático e rápido, pois oferecem diversas ferramentas e funções já prontas e eficientes para todas as etapas, desde a manipulação e análise de dados, construção e treinamento dos modelos de aprendizado de máquina, até a visualização dos resultados obtidos.

2.5.1 NumPy

O NumPy (*Numerical Python*) é uma biblioteca do Python, de código aberto, que oferece suporte para *arrays* (coleções de elementos organizados em posições de memória) e matrizes multidimensionais, além de diversas funções matemáticas de alto desempenho para operar nessas estruturas, como geradores de números aleatórios, rotinas de álgebra linear, transformada de Fourier, entre outras (NUMPY, 2025).

2.5.2 Pandas

O Pandas é uma biblioteca do Python, de código aberto, entre as mais populares para manipulação e análise de dados. Oferece um importante tipo de estruturas de dados, os *dataframes*, que são tabelas bidimensionais com indexação integrada, similares a tabelas de bancos de dados, onde os dados são organizados em linhas e colunas. Permite alinhamento inteligente de dados, tratamento integrado de dados ausentes, remodelagem e dinamização de conjuntos de dados, mesclagem e junção de dados, funcionalidade de séries temporais, como geração de intervalos de datas e conversão de frequência, estatísticas de janela móvel, mudanças de data e atraso, bem como criar deslocamentos de tempo específicos de domínio e unir séries temporais sem perder dados (NUMFOCUS, 2025).

2.5.3 Matplotlib

O Matplotlib é uma biblioteca de código aberto do Python que abrange a criação de visualizações estáticas, animadas e interativas. Engloba criação de gráficos de alta qualidade, figuras interativas e permite personalização do estilo visual e do *layout* (MATPLOTLIB, 2025).

2.5.4 Sklearn

O Scikit-learn (conhecido como sklearn) é uma biblioteca Python de ML (código aberto), muito utilizada para resolver problemas de classificação, regressão, clusterização e redução de dimensionalidade. É construída baseada nas bibliotecas NumPy, SciPy e Matplotlib, tornando-a poderosa e eficiente para análise de dados e modelagem preditiva (SCIKIT-LEARN, 2025).

2.5.5 TensorFlow

O TensorFlow é uma biblioteca Python (código aberto), desenvolvida pela Google, para computação numérica e aprendizado de máquina. Foi projetado para criar, treinar e implementar modelos de aprendizado profundo e outros algoritmos de aprendizado de máquina para uma ampla variedade de dispositivos. O TensorFlow possui vasta flexibilidade, permitindo criar modelos desde os mais simples, como regressões lineares, até os mais complexos, como redes neurais convolucionais e recorrentes (TENSORFLOW, 2025).

3 METODOLOGIA

3.1 Equipamento e softwares

Para o desenvolvimento do projeto foi utilizado um notebook Acer Nitro AN515-52, com processador Intel(R) Core (TM) i7-8750H CPU 2.20Hz (12 CPUs) 2.2GHz, 16GB de memória RAM e sistema operacional Windows 11 Pro.

Foi utilizada a IDE (*Integrated Development Environment*) JupyterLab para visualização, análise e tratamento dos dados, bem como o desenvolvimento dos modelos baseados em redes neurais. O JupyterLab é uma escolha interessante pois permite trabalhar com as linguagens de programação Python, Ruby, Julia, Perl, Matlab e R. Dessa forma, permite a importação de bibliotecas essenciais para a realização de análise exploratória de dados, ciência de dados, computação científica, jornalismo computacional e *machine learning*.

3.2 Conjunto de dados

O conjunto de dados utilizado no projeto abrange dez séries temporais distintas, cada uma correspondendo a um material específico utilizado na cadeia produtiva de carrocerias. Cada série representa a quantidade consumida (em unidades) do respectivo material ao longo do tempo e possui 100 amostras no tempo. As séries temporais possuem granularidade semanal, ou seja, cada amostra reflete o consumo registrado em uma determinada semana. Esses dados foram extraídos de uma empresa do ramo de fabricação de carrocerias.

Foram coletadas 100 amostras semanais, abrangendo um período total de 1 ano, 11 meses e 2 semanas. Esse intervalo foi determinado pela disponibilidade de dados no banco de dados da empresa. Foram selecionados dez materiais que possuem baixa variação de consumo por semana, de acordo com os profissionais responsáveis pelo controle de estoque desses materiais.

3.3 Métodos e procedimentos

Para o desenvolvimento do projeto, foi realizada uma análise exploratória do conjunto de dados afim de compreender estatisticamente o comportamento das séries temporais. A partir da apreciação dos dados, eles passaram por um processo de normalização para um treinamento mais eficiente dos modelos de previsão. Foi feita uma nova análise exploratória nos dados normalizados como validação do processo efetuado. Em seguida, os dados foram preparados e estruturados para a modelagem das redes neurais. Assim, foram construídos diferentes modelos de previsão com o objetivo de comparar seus desempenhos e identificar aquele que melhor se adequa à previsão dos dados de entrada. Os modelos explorados incluem RNN simples, RNN simples com *Batch Normalization*, RNN simples com *Layer Normalization*, LSTM e GRU. Com os resultados dos modelos treinados, foram calculadas as métricas de erro para cada item e modelo, bem como geradas as curvas de aprendizado, que serviram de base para análise e comparação de desempenho.

No Apêndice A, encontra-se a figura com o fluxograma da metodologia utilizada no desenvolvimento do trabalho.

3.3.1 Preparação e estruturação dos dados para modelagem

Inicialmente, foram aplicadas técnicas de estatística descritiva nas dez séries temporais, com o objetivo de realizar uma análise exploratória inicial. Foram calculadas medidas como valor máximo, valor mínimo, média, mediana, desvio padrão, coeficiente de variação, assimetria e curtose, permitindo a caracterização do comportamento individual de cada série.

Após essa etapa, os dados foram submetidos a um processo de normalização por meio da classe MinMaxScaler, da biblioteca Scikit-learn, ajustando os valores para o intervalo entre 0 e 1. Em seguida, foi realizada uma nova rodada de estatística descritiva com os dados normalizados, visando avaliar a homogeneização das escalas e confirmar a adequação das séries temporais para aplicação em modelos preditivos.

As séries temporais normalizadas foram carregadas para um *dataframe* da biblioteca Pandas do Python, que por sua vez foi convertido em um *array* da biblioteca NumPy, no formato (10, 100, 1), ou seja, um *array* com 10 elementos na primeira dimensão (quantidade de séries temporais distintas), 100 elementos ao longo da segunda dimensão (quantidade de

amostras temporais para cada elemento da primeira dimensão) e 1 nível de profundidade (cada amostra de cada série possui apenas uma quantidade de consumo). Essa estrutura de dados facilita a divisão de conjuntos de treino, validação e teste.

A partir do *array* criado, foi realizada a separação das bases de treino, validação e teste, garantindo não só que os modelos aprendam os padrões presentes nos dados históricos, mas também sejam capazes de generalizar para dados futuros não vistos, evitando o sobreajuste. Cada uma das 10 séries temporais foi particionada utilizando uma técnica de janelas deslizantes para os modelos que preveem um passo à frente.

As primeiras 70 semanas de dados de consumo de cada item (índices 0 a 69) foram reservadas para a criação das janelas de treinamento. As semanas subsequentes (índices 70 a 79) foram utilizadas para criar as janelas do conjunto de validação. Já as últimas 20 semanas de dados de cada item (índices 80 a 99) foram reservadas para o teste final do desempenho dos modelos.

A técnica de janelas deslizantes foi empregada definindo inicialmente um tamanho de janela de entrada (15 semanas). O conjunto de treino de entrada (X_train) consiste em sequências de 15 semanas consecutivas extraídas dos dados de treino (amostras de 0 a 69). Por exemplo, a primeira janela seria das semanas 0 a 14, a segunda das semanas 1 a 15, e assim por diante. O conjunto de treino de saída (y_train) traz o valor alvo correspondente à cada sequência do X_train. Por exemplo, para a janela de entrada das semanas 0-14, o y_train alvo foi o consumo da semana 15.

O conjunto de validação de entrada (X_valid) foi criado de forma similar, utilizando as janelas de 15 semanas que antecedem o período de validação. Por exemplo, para prever o consumo da semana 70, a janela de entrada em X_valid foi das semanas 55 (70 – 15, sendo 15 o tamanho da janela) até 69. Já o conjunto de validação de saída (y_valid) foram os valores alvo correspondentes, ou seja, o valor de consumo real das semanas de 70 a 79. O conjunto de validação (X_valid e y_valid) foi utilizado no método *fit* do modelo com intuito de monitorar o desempenho em dados não vistos durante o treinamento.

Os conjuntos de treino e validação de entrada (X_train e X_valid) foram formatados como *arrays* tridimensionais (número de janelas, tamanho da janela, 1) e os conjuntos de treino e validação de saída (y_train e y_valid) como *arrays* bidimensionais (número de janelas, 1), adequados para a entrada dos modelos de redes neurais.

Já o conjunto de teste foi gerado utilizando a abordagem de previsão recursiva para avaliar a capacidade dos modelos de prever os 20 passos futuros (semanas 80 a 99). Uma sequência inicial de entrada correspondente às últimas 15 semanas antes do período de teste (semanas 65 a 79) foi fornecida ao modelo treinado. Após o modelo prever o consumo para a próxima semana (semana 80), o valor previsto é incorporado à sequência de entrada (removendo o valor mais antigo) para então prever o consumo da semana 81. Esse processo iterativo seguiu-se até que todas as 20 semanas (80 a 99) fossem previstas.

As previsões geradas foram comparadas com os valores reais de consumo dessas 20 semanas a fim de calcular as métricas de erro após a reversão dos dados para sua escala original.

Para otimizar o processo de treinamento dos modelos e principalmente prevenir o sobreajuste, foi empregado a ferramenta EarlyStopping, disponibilizada pela biblioteca do TensorFlow. Esta técnica consiste em monitorar uma métrica de desempenho do modelo em um conjunto de dados de validação durante o treinamento.

A EarlyStopping foi configurada para observar a perda no conjunto de validação (*val_loss*). Se essa métrica não apresentasse melhoria (ou seja, diminuição) por um número pré-definido de épocas consecutivas (parâmetro *patience*), o treinamento era automaticamente interrompido. Além disso, utilizou-se o parâmetro *restore_best_weights* como verdadeiro, garantindo que ao final do processo os pesos do modelo fossem revertidos para aqueles correspondentes à época de melhor desempenho no conjunto de validação.

Ou seja, a utilização do EarlyStopping visa identificar o momento em que o modelo atinge sua melhor capacidade de generalização para dados não vistos, evitando que o treinamento prossiga desnecessariamente e que o modelo comece a se especializar excessivamente nos dados de treino, prejudicando seu desempenho em previsões de dados não vistos.

Para controlar a aleatoriedade dos resultados no projeto, foram definidas as sementes aleatórias para as bibliotecas NumPy e TensorFlow. A semente aleatória define um ponto inicial fixo para a geração de números aleatórios, garantindo assim a reprodutibilidade dos resultados em diferentes execuções do código. Caso não seja definida, a inicialização dos pesos nas redes neurais será aleatória, onde cada execução poderia obter resultados diferentes. Além disso, a definição das sementes aleatórias garante que todos os modelos comecem do mesmo estado inicial, evitando com que as diferenças de desempenho sejam causadas pela aleatoriedade.

3.3.2 Compilação e treinamento dos modelos

Todos os modelos do presente projeto foram compilados utilizando a função de perda MSE (erro quadrático médio), pois sua utilização é ampla em tarefas de previsão e por penalizar fortemente valores discrepantes. Foi escolhido o otimizador Adam, da biblioteca TensorFlow, com uma taxa de aprendizado inicial de 0.001 (valor comumente recomendado por pesquisadores da área), devido sua eficiência e bom desempenho em diversos problemas de aprendizado profundo, principalmente em cenários com dados ruidosos e aprendizado não linear. O otimizador ajusta eficientemente os pesos dos modelos durante a fase de treinamento, buscando minimizar a função do erro.

Já a métrica MAE (erro médio absoluto) foi monitorada durante os treinamentos, visando fornecer uma medida adicional do erro de previsão na mesma unidade dos dados originais.

Os modelos foram treinados com os conjuntos X_train e y_train (preparados com janelas deslizantes de 15 semanas para prever o próximo passo) e validados com os conjuntos X_valid e y_valid.

Para prevenção de sobreajuste e seleção do modelo com melhor capacidade de generalização em cada técnica, foi utilizada a ferramenta EarlyStopping para monitorar a perda no conjunto de validação e interromper o treinamento caso não ocorra melhoria nessa métrica por 15 épocas consecutivas (*patience = 15*). O treinamento foi configurado para um máximo de 200 épocas (*epochs = 200*), sendo que a parada efetiva pode ocorrer antes das 200 épocas através do EarlyStopping.

Após o treinamento, as curvas de aprendizado (perda de treino e perda de validação ao longo das épocas) foram geradas para análise visual do processo de treinamento. Em seguida, o modelo já treinado, foi utilizado para realizar as previsões recursivas no conjunto de teste com objetivo de realizar a avaliação final do seu desempenho através do cálculo das métricas de erro a partir das previsões geradas.

3.3.3 Modelo RNN com camadas simples

A primeira rede implementada foi uma rede neural recorrente simples, utilizando a ferramenta Keras da biblioteca TensorFlow do Python. A estrutura do modelo é sequencial, cujas camadas são empilhadas linearmente uma após a outra. O modelo foi construído a partir de duas camadas simples (SimpleRNN), seguidas de uma camada densa (*Dense*), responsável por transformar as representações aprendidas pelas camadas recorrentes anteriores na previsão final de consumo.

A primeira camada SimpleRNN foi configurada com 20 neurônios e com o parâmetro *return_sequences = True*, indicando que a camada retorna uma sequência de saídas para cada passo de tempo da janela de entrada ao invés de apenas o último valor, uma vez que a próxima camada também espera uma sequência de valores em sua entrada. O parâmetro *input_shape = [15, 1]*, sendo 15 o tamanho da janela (número de semanas anteriores de consumo utilizadas para prever a próxima) e 1 indica apenas uma variável (quantidade de consumo) sendo utilizada.

A próxima camada SimpleRNN também possui 20 neurônios, mas define o parâmetro *return_sequences = False*, fazendo com que a camada retornasse apenas o último valor de saída da sequência, representando o resumo da informação temporal aprendida até o momento.

A quantidade de neurônios de ambas as camadas foi definida empiricamente com base em testes que equilibram o desempenho preditivo e o custo computacional.

A camada final (camada Dense) possui apenas um neurônio, adequado para previsões de séries temporais univariadas. Ela realiza uma combinação linear da saída da última camada simples para produzir a previsão final.

A compilação e treinamento do modelo foram realizadas conforme o procedimento descrito na seção 3.3.2.

3.3.4 Modelo RNN simples com *Batch Normalization*

A segunda rede implementada foi uma rede neural recorrente simples (SimpleRNN) com *Batch Normalization* (normalização em lote), uma técnica que visa melhorar a velocidade, desempenho e estabilidade do treinamento de redes neurais. O modelo foi construído a partir de duas camadas simples, duas camadas de BatchNormalization e uma camada densa, também importadas da biblioteca TensorFlow.

A primeira camada RNN foi configurada com 20 neurônios, parâmetro *return_sequences = True* e o parâmetro *input_shape = [15, 1]*, da mesma maneira como foi construída a primeira camada do modelo de RNN simples.

A segunda camada BatchNormalization foi aplicada à saída sequencial da primeira camada SimpleRNN, sem configuração adicional, ou seja, com seus parâmetros padrão.

A terceira camada SimpleRNN foi aplicada em sequência e configurada com 20 neurônios e parâmetro *return_sequences = False*, da mesma forma que a segunda camada do modelo de RNN simples.

A quarta camada BatchNormalization foi aplicada em sequência, também com seus parâmetros padrão.

Por fim, a quinta camada foi aplicada após a quarta camada, uma camada Dense com apenas um neurônio, da mesma forma que fora aplicada no modelo de RNN simples.

A compilação e treinamento do modelo foram realizadas conforme o procedimento descrito na seção 3.3.2.

3.3.5 Modelo RNN simples com *Layer Normalization*

Visando aprimorar a estabilidade e o desempenho do treinamento das redes recorrentes, no terceiro modelo implementado foi utilizada uma célula RNN Simples customizada, que incorpora a técnica de *Layer Normalization* (normalização de camada). Essa técnica normaliza as entradas dentro de cada camada de forma independente para cada amostra e para cada passo no tempo, atuando sobre as variáveis alvo.

Para implementação da técnica no modelo foi definida uma célula RNN customizada, denominada LNSimpleRNNCell. A célula encapsula uma SimpleRNNCell padrão, mas com sua função de ativação interna configurada como None. Após a operação da SimpleRNNCell interna, uma cama LayerNormalization é aplicada à saída. A função de ativação desejada (por exemplo, tangente hiperbólica) é aplicada à saída normalizada pela camada anterior, garantindo que a normalização ocorra antes da não-linearidade da ativação para cada passo de tempo.

O modelo sequencial de camadas foi construído utilizando esta célula customizada através da camada RNN do TensorFlow. A primeira camada é uma camada RNN utilizando a célula LNSimpleRNNCell com 20 unidades. O parâmetro *return_sequences = True* foi definido para que a camada retornasse à sequência completa de saídas normalizadas para a próxima camada. O parâmetro *input_shape = [15,1]* foi configurado definindo o tamanho da janela e o número de variáveis.

A segunda camada foi aplicada em sequência, sendo também uma camada RNN com células LNSimpleRNNCell de 20 neurônios. Nesta camada, o parâmetro *return_sequences =*

False foi utilizado de modo que apenas a saída do último passo de tempo (após normalização e ativação) passasse para a próxima camada.

A terceira e última camada foi uma camada Dense com um único neurônio para produzir a previsão final.

A compilação e treinamento do modelo foram realizadas conforme o procedimento descrito na seção 3.3.2.

3.3.6 Modelo LSTM

Para lidar com possíveis dependências de longo prazo nas séries temporais e mitigar o problema do gradiente evanescente comum em RNNs simples, o quarto modelo implementado foi baseado em células LSTM. Ele foi construído com camadas sequenciais, sendo a primeira camada LSTM, a segunda camada Dropout, a terceira camada também LSTM, a quarta camada também Dropout e a quinta camada Dense.

A primeira camada LSTM da biblioteca TensorFlow foi configurada com 40 unidades. Os parâmetros *return_sequences = True* e *input_shape = [15,1]* foram utilizados de maneira similar à camada SimpleRNN dos modelos anteriores. Foi aplicado o parâmetro *recurrent_dropout* de 0,2 com intuito de regularizar as conexões recorrentes dentro das células LSTM e mitigar o sobreajuste específico das dependências temporais.

A segunda camada Dropout (biblioteca TensorFlow) foi configurada com uma taxa de 0,2 e adicionada após a primeira camada LSTM, realizando a regularização das conexões e auxiliando na prevenção do sobreajuste.

A terceira camada adicionada em sequência foi outra camada LSTM, também com 40 unidades e *recurrent_dropout* de 0,2. O parâmetro *return_sequences = False* foi configurado para retorno de apenas a saída do último passo de tempo da sequência.

A quarta camada foi outra camada de Dropout, configurada com uma taxa de 0,2.

A quinta e última camada foi uma camada Dense de uma única unidade, produzindo a previsão de consumo para o próximo passo de tempo.

As taxas de Dropout e os valores de *recurrent_dropout* foram definidos empiricamente com base em testes que equilibravam o desempenho preditivo e o custo computacional.

A compilação e treinamento do modelo foram feitas conforme descrito na seção 3.3.2.

3.3.7 Modelo GRU

O quinto modelo implementado foi baseado em camadas GRU. Assim como as LSTMs, elas foram projetadas para superar as limitações das RNNs simples na captura de dependências de longo prazo e no manejo do problema do gradiente evanescente, mas o fazem com uma arquitetura de célula um pouco mais simplificada.

A implementação do modelo utilizou camadas GRU envolvidas por uma camada envoltória chamada Bidirectional (biblioteca TensorFlow) permitindo que a rede processasse as sequências de entrada em ambas as direções (passado para o futuro e futuro para o passado) para capturar um contexto mais abrangente das séries temporais.

O modelo foi construído com sua primeira camada sendo uma camada GRU Bidirectional, uma segunda camada de Dropout, uma terceira GRU Bidirectional, uma quarta Dropout e uma quinta e última camada Dense.

A primeira camada foi configurada a partir de camada Bidirectional envolvendo uma camada GRU com 30 unidades. Parâmetros *return_sequences = True* e *input_shape = [15,1]* foram utilizados de maneira similar à camada SimpleRNN dos modelos anteriores. Foi aplicado um *recurrent_dropout* de 0,1 para regularização das conexões recorrentes.

A segunda camada Dropout foi configurada com uma taxa de 0,2 e adicionada após a primeira camada Bidirectional, realizando a regularização das conexões e auxiliando na prevenção do sobreajuste.

A terceira camada adicionada foi mais uma camada Bidirectional composta por uma camada GRU com 40 unidades e *recurrent_dropout* de 0,1. O parâmetro *return_sequences = False* foi configurado para retorno da saída do último passo de tempo.

A quarta camada foi outra camada de Dropout configurada com uma taxa de 0,2. A quinta e última camada foi uma camada Dense de uma única unidade, produzindo a previsão de consumo para o próximo passo de tempo.

Com base em testes que equilibravam o desempenho preditivo e o custo computacional, as taxas de Dropout e os valores de *recurrent_dropout* foram definidos empiricamente.

A compilação e treinamento do modelo foram feitas conforme descrito na seção 3.3.2.

3.3.8 Análise e discussão dos resultados

A avaliação do desempenho dos modelos foi realizada através de duas abordagens complementares. Primeiramente, para uma avaliação quantitativa, foram calculadas três métricas de erro distintas sobre o conjunto de teste, composto pelas últimas 20 semanas de dados de consumo.

O Erro Médio Absoluto (MAE) foi utilizado para obter uma medida direta da magnitude média do erro, expressa na mesma unidade do material (quantidade de consumo), ou seja, quanto, em média, os valores previstos se distanciam dos valores reais. A Raiz do Erro Quadrático Médio (RMSE), por sua vez, foi analisada em comparação com o MAE, pois uma diferença significativa entre os dois indica a presença de erros de grande magnitude, dado que o RMSE penaliza mais severamente desvios maiores. Por fim, o Erro Percentual Absoluto Médio (MAPE) foi empregado para avaliar a acurácia relativa, permitindo uma comparação de desempenho mais equilibrado entre itens com diferentes escalas de consumo, visto que o MAPE é uma métrica percentual.

Em segundo lugar, as curvas de aprendizado foram geradas ao final dos treinamentos, com apoio da biblioteca Matplotlib, plotando a perda (MSE) dos conjuntos de treino e validação por época para diagnosticar o comportamento do treinamento, a convergência e o sobreajuste.

Com apoio da biblioteca NumPy, o cálculo das métricas de erro foi realizado comparando as previsões com os valores reais. Como os modelos foram treinados para prever apenas um passo à frente, as 20 previsões de teste foram geradas através da estratégia de previsão recursiva descrita anteriormente. Para garantir a interpretabilidade dos resultados, tanto as previsões quanto os dados reais foram revertidos para sua escala original antes do cálculo dos erros. As métricas foram apuradas para cada um dos dez itens individualmente, e uma média foi então calculada para se obter o desempenho agregado de cada arquitetura de modelo.

Por fim, foi realizada uma análise aprofundada e comparativa do desempenho dos modelos de previsão desenvolvidos com objetivo de interpretar criticamente os resultados, correlacionando a *performance* de cada arquitetura com seus fundamentos teóricos e com as características inerentes às séries temporais de consumo. Desta forma, a discussão busca esclarecer os possíveis motivos que levaram cada modelo a apresentar seu respectivo desempenho, culminando em uma classificação da sua eficácia e adequação para a tarefa de previsão de demanda de materiais neste contexto industrial.

4 RESULTADOS E DISCUSSÕES

Nesta seção, são apresentados e analisados os resultados obtidos em cada etapa do projeto, desde a caracterização inicial dos dados até a avaliação comparativa dos modelos de previsão desenvolvidos.

4.1 Análise exploratória do conjunto de dados de consumo

Antes da implementação e treinamento dos modelos de previsão, foi realizada uma análise exploratória das dez séries temporais de consumo de materiais. As estatísticas descritivas aplicadas compreendem valores máximos e mínimos, médias, medianas, desvios padrão, coeficientes de variação, assimetrias, curtoses e variâncias. A Tabela 1 apresenta os resultados desses cálculos.

Tabela 1 - Estatísticas descritivas das séries temporais

Item	Máx.	Mín.	Média	Mediana	Desvio padrão	Coeficiente variação	Assimetria	Curtose
1	213	186	200,25	199,0	7,5229	3,7567	0,0071	-1,0982
2	87	73	79,45	79,0	3,5573	4,4773	0,3127	-0,8469
3	109	91	100,23	100,5	5,0309	5,0194	-0,0730	-0,7862
4	124	117	119,82	120,0	1,9404	1,6195	0,2472	-1,1781
5	71	49	59,59	58,0	6,1809	10,3724	0,1343	-1,0932
6	151	130	140,29	141,0	5,8712	4,1850	-0,0730	-1,0698
7	193	167	180,20	180,0	7,0022	3,8858	-0,1132	-1,0344
8	164	136	151,09	151,0	7,6846	5,0861	-0,0901	-1,0368
9	241	220	229,12	228,0	6,0357	2,6343	0,2955	-0,9887
10	127	113	120,76	122,0	3,5820	2,9662	-0,2191	-1,0080

Fonte: Autoria própria.

Analisando os valores máximos e mínimos dos dez itens, foi observado que as séries apresentam flutuações consideravelmente distintas. Por exemplo, o item 5 tem um valor máximo de 71 unidades, enquanto o item 9 possui um valor máximo de 241 unidades. Essa discrepância sugere a necessidade de normalização das séries temporais.

Ao analisar a média e a mediana dos dez itens, foi analisado que seus valores são bastante próximos se comparados à magnitude dos dados. Isso sugere que as séries temporais possuem distribuições simétricas ou quase simétricas, e baixa presença valores extremos (*outliers*), o que facilita o treinamento de redes neurais.

O desvio padrão e o coeficiente de variação são métricas importantes para avaliar a dispersão dos dados em torno da média. Ao analisar os valores obtidos, foi verificado que o desvio padrão dos itens esteve, na maioria dos casos, entre 5 e 8 unidades. No entanto, ao examinar o coeficiente de variação, percebeu-se uma dispersão percentual mais evidente, com valores que variam de 1,62% a 10,37%. Isso ocorre porque, embora os desvios padrão sejam relativamente próximos, as médias das séries temporais apresentam diferenças mais expressivas. Ainda assim, o coeficiente de variação mais elevado de 10,37%, indica uma variabilidade baixa, caracterizando as dez séries temporais como consumos controlados e relativamente previsíveis.

Ao analisar a assimetria das séries temporais dos itens, foi verificado um intervalo de valores entre -0,219 e 0,312, indicando uma leve assimetria. Isso sugere que as séries possuem distribuições próximas da simetria, o que tende a facilitar o aprendizado em modelos preditivos de redes neurais.

Analisando os valores de curtose das séries temporais dos itens, foi observado um intervalo de valores entre -1,18 e -0,79, sugerindo que todas possuem distribuições platicúrticas, ou seja, curtoses abaixo de zero. Esse tipo de distribuição é caracterizado por um achatamento em relação à curva normal, indicando menores concentrações de valores em torno da média, e caudas menos longas. O achatamento da distribuição demonstra uma variabilidade mais homogênea entre os valores das séries, e as caudas menos longas indicam a ausência de valores extremos favorecendo o uso de modelos preditivos com menor risco de sobreajuste devido a *outliers*.

Após a análise estatística descritiva, concluiu-se que, individualmente, as séries temporais apresentam comportamentos favoráveis para aplicação em modelos preditivos. No entanto, ao considerar as dez séries em conjunto, observou-se a necessidade de normalização, uma vez que os valores máximos, mínimos e as médias, apresentam magnitudes consideravelmente distintas, o que pode reduzir o desempenho e a convergência do modelo. Para isso, utilizou-se a biblioteca sklearn para importar a classe MinMaxScaler e aplicar sua técnica de normalização, ajustando os dados para um intervalo específico, entre 0 e 1.

Após a normalização, foi realizada uma nova análise estatística descritiva, onde foi identificada menor variabilidade dos valores máximos, mínimos e de médias entre as séries temporais dos materiais, objetivo principal da técnica aplicada. A Tabela 2 apresenta as estatísticas das séries temporais normalizadas.

Tabela 2 - Estatísticas descritivas das séries temporais normalizadas

Item	Máx.	Mín.	Média	Mediana	Desvio padrão	Coeficiente variação	Assimetria	Curtose
1	1	0	0,5278	0,4815	0,2786	52,7920	0,0071	-1,0982
2	1	0	0,4607	0,4286	0,2541	55,1512	0,3127	-0,8469
3	1	0	0,5128	0,5278	0,2795	54,5062	-0,0730	-0,7862
4	1	0	0,4029	0,4286	0,2772	68,8094	0,2472	-1,1781
5	1	0	0,4814	0,4091	0,2810	58,3658	0,1343	-1,0932
6	1	0	0,4900	0,5238	0,2796	57,0570	-0,0730	-1,0698
7	1	0	0,5077	0,5000	0,2693	53,0467	-0,1132	-1,0344
8	1	0	0,5389	0,5357	0,2744	50,9248	-0,0901	-1,0368
9	1	0	0,4343	0,3810	0,2874	66,1811	0,2955	-0,9887
10	1	0	0,5543	0,6429	0,2559	46,1598	-0,2191	-1,0080

Fonte: Autoria própria.

O desvio padrão dos itens apresentou valores mais próximos entre eles, variando de 0,2541 a 0,281, demonstrando que a variação em torno da média abaixou consideravelmente e as séries possuem um comportamento mais parecido entre elas. O aumento do coeficiente de variação, que passou a ter valores de 50,9% até 68,8%, se justifica, pois, a média das séries passou a apresentar valores menores que um e próximos de zero. Como a média é o denominador do cálculo do coeficiente de variação, os percentuais sobem drasticamente e o coeficiente se torna menos interpretável após a normalização, apesar de não significar de fato aumento na variabilidade dos dados. Nesse cenário, é recomendável analisar apenas o desvio padrão.

A assimetria e a curtose das séries não se alteraram após a normalização, uma vez que são dados estatísticos adimensionais, ou seja, não dependem da escala das séries, apenas da forma da distribuição. Como a normalização aplicada transforma as séries para um intervalo específico, mas preserva a proporção entre os valores, a forma da distribuição permanece inalterada.

A análise estatística descritiva das séries temporais normalizadas revelou que, após a normalização, as mesmas apresentam um comportamento favorável para aplicação em modelos preditivos.

4.2 Desempenho do modelo RNN Simples

A partir das quantidades de consumo dos itens, especificamente das 20 últimas semanas (semana 81 a 100) e dos valores resultantes das previsões dessas mesmas semanas, foram calculadas as métricas de erro para cada item. Os resultados estão apresentados na Tabela 3.

Tabela 3 - Métricas de erro por item do modelo RNN Simples

Item	MAE	RMSE	MAPE (%)
Item 1	2,610	3,083	1,317
Item 2	1,490	1,979	1,815
Item 3	1,738	1,904	1,656
Item 4	0,642	0,774	0,538
Item 5	1,996	2,419	3,702
Item 6	4,750	5,652	3,476
Item 7	3,188	3,868	1,744
Item 8	1,144	1,380	0,731
Item 9	1,046	1,278	0,468
Item 10	2,961	3,808	2,402

Fonte: Autoria própria.

Para obter as métricas de erro finais do modelo, calculou-se a média de cada uma das métricas avaliadas. Estes resultados consolidados são detalhados na Tabela 4.

Tabela 4 - Métricas de erro finais do modelo RNN Simples

Modelo	MAE	RMSE	MAPE (%)
RNN Simples	2,157	2,614	1,785

Fonte: Autoria própria.

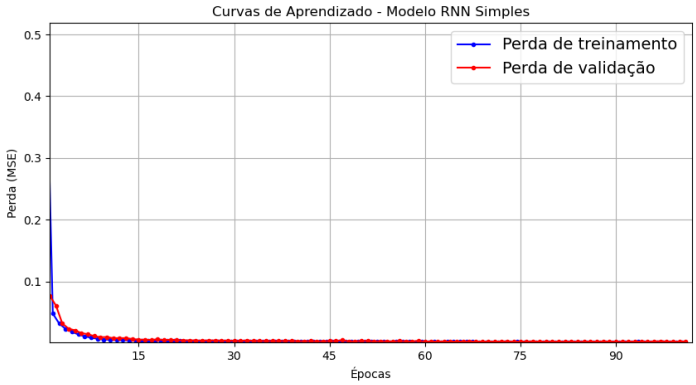
O modelo de RNN simples apresentou um ótimo desempenho geral, com baixos valores em todas as três métricas. Analisando as métricas para cada item, observa-se uma *performance* geral muito boa, com 7 dos 10 itens apresentando um MAPE abaixo de 2%. Os itens 5 e 6 apresentaram erros um pouco maiores, mas ainda percentualmente baixos, com MAPE de 3,70% e 3,48%, respectivamente. A diferença entre RMSE e MAE para a maioria dos itens não é consideravelmente significativo, sugerindo que não há predominância de erros muito discrepantes, mas que há uma certa consistência nos erros de previsão.

Analisando as métricas finais para este modelo, observamos que ele alcançou um MAPE médio de 1,78%, evidenciando alta precisão, com previsões médias que se aproximaram

significativamente dos valores reais de consumo. Além disso, a proximidade entre os valores de MAE e RMSE indica uma baixa ocorrência de erros discrepantes, como sugere a análise das métricas por cada item.

O gráfico de curva de aprendizado contendo a curva de perda de treino e a curva de perda de validação ao longo das épocas foi gerada e apresentada na Figura 9.

Figura 9 - Curvas de aprendizado do modelo RNN Simples



Fonte: Autoria própria.

A curva de aprendizado deste modelo revelou um treinamento eficiente. As perdas de treino e validação diminuíram rapidamente nas primeiras 15 épocas, estabilizando-se em seus valores mínimos a partir desse ponto. O EarlyStopping selecionou o modelo na época 101, mas analisando a curva, a interrupção do treinamento antes dessa época poderia ter melhor capacidade de generalização.

Em termos gerais, a simplicidade do modelo pode ter sido uma vantagem diante do conjunto de dados de entrada visto que os resultados são promissores.

4.3 Desempenho do modelo RNN Simples com *Batch Normalization*

A partir das quantidades de consumo dos itens das 20 últimas semanas (semana 81 a 100) e dos valores resultantes das previsões dessas semanas, foram calculados os erros para cada item. Os resultados são apresentados na Tabela 5.

Tabela 5 - Métricas de erro por item do modelo RNN com *Batch Normalization*

Item	MAE	RMSE	MAPE (%)
Item 1	3,532	3,946	1,774
Item 2	1,055	1,277	1,292
Item 3	4,570	6,057	4,302
Item 4	0,595	0,749	0,501
Item 5	8,516	9,993	15,836
Item 6	4,406	6,879	3,283
Item 7	7,363	8,590	4,102
Item 8	4,923	6,043	3,135
Item 9	1,286	1,632	0,570
Item 10	3,137	3,975	2,538

Fonte: Autoria própria.

Para obter as métricas de erros finais do modelo, calculou-se a média de cada uma das métricas avaliadas. Estes resultados consolidados são detalhados na Tabela 6.

Tabela 6 - Métricas de erro finais do modelo RNN com *Batch Normalization*

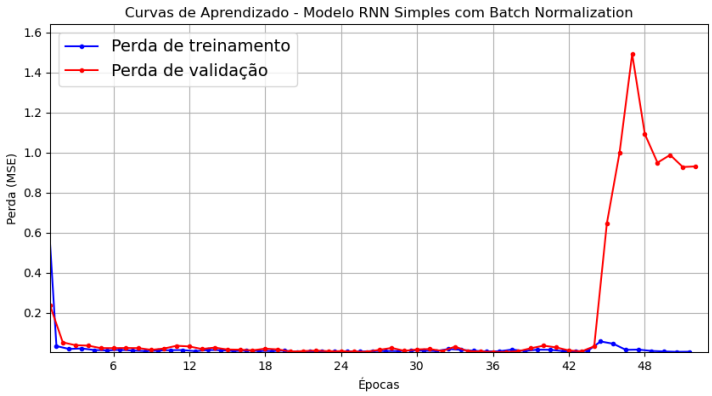
Modelo	MAE	RMSE	MAPE (%)
RNN com *Batch Normalization*	3,938	4,914	3,733

Fonte: Autoria própria.

O modelo de RNN com *Batch Normalization* apresentou um desempenho mais baixo em relação ao desempenho do modelo de RNN simples, com erros consideravelmente mais altos. Analisando as métricas para cada item, tivemos um aumento geral nos valores com a aplicação da técnica de BN, com o item 5 chegando ao percentual MAPE de 15,84%. A média final das métricas também apresentou valores mais altos de MAE, RMSE e MAPE, com MAPE de 3,73%, sugerindo maior erro absoluto, discrepância e erro percentual entre valores reais e previstos.

O gráfico de curva de aprendizado contendo a curva de perda de treino e a de perda de validação ao longo das épocas foi gerado e é mostrado na Figura 10.

Figura 10 - Curvas de aprendizado do modelo RNN com *Batch Normalization*



Fonte: Autoria própria.

A curva de aprendizado do modelo RNN simples com BN sugere uma instabilidade no treinamento. A perda de treinamento começa alta, cai rapidamente para um nível baixo e permanece assim, com algumas flutuações. Já a perda de validação apresenta um comportamento bastante instável e ruidoso. Embora atinja valores baixos em certos momentos, ela exibe picos significativos em várias outras épocas. Isso indica que o desempenho do modelo em dados não vistos varia muito de uma época para outra durante o treinamento.

Em termos gerais, a aplicação da técnica de *Batch Normalization* com os parâmetros configurados demonstrou não ser ideal para o conjunto de dados de entrada, resultando em baixa capacidade de generalização do modelo. A instabilidade também sugere ter dificultado para que o EarlyStopping pudesse determinar um ponto ótimo de parada do treinamento.

4.4 Desempenho do modelo RNN Simples com *Layer Normalization*

A partir das quantidades de consumo dos itens das 20 últimas semanas (semana 81 a 100) e dos valores resultantes das previsões dessas mesmas semanas, foram calculadas as métricas de erro para cada item. Os resultados estão exibidos na Tabela 7.

Para obter as métricas de erro finais do modelo, calculou-se a média de cada uma das métricas avaliadas. Estes resultados consolidados são detalhados na Tabela 8.

Tabela 7 - Métricas de erro por item do modelo RNN com *Layer Normalization*

Item	MAE	RMSE	MAPE (%)
Item 1	2,648	3,026	1,345
Item 2	1,550	1,844	1,900
Item 3	3,510	3,809	3,345
Item 4	1,506	1,718	1,270
Item 5	4,816	5,583	8,841
Item 6	5,510	6,476	4,065
Item 7	3,907	4,685	2,171
Item 8	4,030	4,642	2,581
Item 9	2,846	3,163	1,265
Item 10	1,922	2,439	1,556

Fonte: Autoria própria.

Tabela 8 - Métricas de erro finais do modelo RNN com *Layer Normalization*

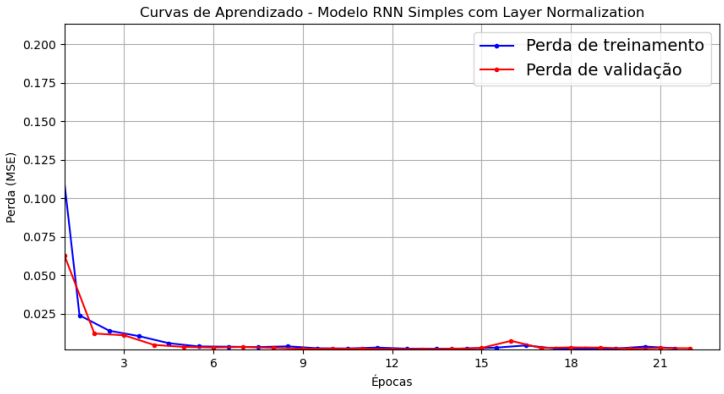
Modelo	MAE	RMSE	MAPE (%)
RNN com *Layer Normalization*	3,224	3,738	2,833

Fonte: Autoria própria.

O modelo de RNN com *Layer Normalization* apresentou um bom desempenho quantitativo, demonstrando ser melhor que a técnica de *Batch Normalization* para esse conjunto de dados, mas não o suficiente a ponto de superar o desempenho da RNN simples. O resultado por item das métricas apresentou valores baixos para diversos itens, mas consideravelmente altos para outros, como por exemplo para o item 5, que alcançou um MAPE de 8,84%. O resultado final mostra o segundo maior MAPE (2,83%) até o momento, representando a influência de valores de MAPE altos na previsão de alguns itens, apesar de 5 dos 10 itens apresentarem MAPE entre 1% e 2%. Os valores de MAE e RMSE foram relativamente próximos entre si, sugerindo baixa presença de valores discrepantes nas previsões.

A Figura 11 apresenta o gráfico de curvas de aprendizado contendo a curva de perda de treino e a de perda de validação ao longo das épocas.

Figura 11 - Curvas de aprendizado do modelo RNN com *Layer Normalization*



Fonte: Autoria própria.

A curva de aprendizado desse modelo demonstra uma rápida capacidade de aprendizado, com ambas as perdas caindo para valores baixos em poucas épocas (em torno de 5 a 7 épocas), sugerindo que a técnica pode estar auxiliando na convergência inicial do modelo. Após atingir o valor mínimo, ambas as perdas apresentam oscilações, com tendências de subida em certos pontos. Isso pode ser observado entre as épocas 15 e 18, entretanto não apresenta um sobreajuste severo, mas demonstra certa dificuldade na generalização dos dados. O treinamento ter ocorrido até a época 22 sugere que o EarlyStopping fez a interrupção enquanto as oscilações não representaram uma piora significativa no desempenho do modelo, auxiliando na obtenção de um ponto de parada de treinamento um pouco melhor.

Em termos gerais, a implementação da técnica de *Layer Normalization* se mostrou mais eficiente em relação a de *Batch Normalization*, mas ainda assim não apresentou resultados superiores ao emprego de apenas a RNN simples. O modelo mostra um aprendizado rápido, mas apresenta o desafio de estabilizar a perda de validação ao longo das épocas, dificultando o alcance de uma generalização mais robusta.

4.5 Desempenho do modelo LSTM

Da mesma forma, considerando as quantidades de consumo dos itens das 20 últimas semanas (semana 81 a 100) e dos valores resultantes das previsões dessas semanas, as métricas de erro para cada item foram calculadas. Os resultados estão apresentados na Tabela 9.

Tabela 9 - Métricas de erro por item do modelo LSTM

Item	MAE	RMSE	MAPE (%)
Item 1	0,790	1,034	0,395
Item 2	0,832	1,015	1,017
Item 3	3,781	4,424	3,573
Item 4	0,972	1,180	0,820
Item 5	4,912	5,907	9,160
Item 6	3,543	4,020	2,600
Item 7	4,686	5,543	2,588
Item 8	4,210	5,167	2,659
Item 9	2,833	3,066	1,263
Item 10	2,889	3,629	2,339

Fonte: Autoria própria.

Em seguida, calculou-se a média de cada uma das métricas avaliadas e cujos resultados consolidados estão apresentados na Tabela 10.

Tabela 10 - Métricas de erro finais do modelo LSTM

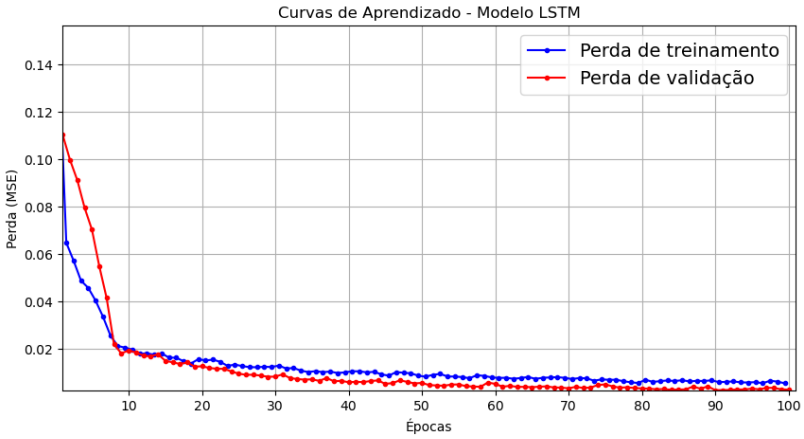
Modelo	MAE	RMSE	MAPE (%)
LSTM	2,826	3,332	2,561

Fonte: Autoria própria.

Este modelo apresentou resultados competitivos em relação aos resultados da RNN com LN. Analisando as métricas de erro para cada item, o MAPE obtido para 4 dos 10 itens está abaixo de 2% e o maior MAPE obtido foi de 9,16% (item 5). As métricas gerais para o modelo apresentaram um MAE de 2,83, um RMSE de 3,33 e um MAPE de 2,56%. Os valores relativamente próximos de MAE e RMSE sugerem baixa presença de valores discrepantes entre previsões e valores reais, e apesar do valor do MAPE ser baixo, o que demonstra um bom desempenho do modelo, o mesmo não superou os resultados da RNN simples.

O gráfico de curvas de aprendizado contendo a curva de perda de treino e a de perda de validação ao longo das épocas foi gerada e apresentada na Figura 12.

Figura 12 - Curvas de aprendizado do modelo LSTM



Fonte: Autoria própria.

A curva de aprendizado do modelo LSTM indicou um bom aprendizado inicial, com as perdas de treinamento e validação diminuindo rapidamente até a 20ª época. Após essa queda inicial, ambas as perdas continuam a diminuir de forma mais lenta e gradual, indicando que o modelo está conseguindo aprender os padrões com os dados de treino e também generalizar esse aprendizado por meio de dados não vistos. A perda de validação se mantém menor que a perda de treinamento, provavelmente devido a regularização aplicada, tornando a tarefa de treinamento mais difícil para o modelo. Em contrapartida, isso melhora a sua capacidade de generalização, resultando em um bom desempenho na validação. Há grandes chances de que sem a regularização, a perda de treinamento apresente sobreajuste, sugerindo um bom emprego da técnica. As curvas aproximam-se de valores consideravelmente baixos até a última época, onde o treinamento foi interrompido pelo EarlyStopping na época 100.

4.6 Desempenho do modelo GRU

A partir das quantidades de consumo dos itens das 20 últimas semanas (semana 81 a 100) e dos valores resultantes das previsões dessas mesmas semanas, foram calculadas as métricas para cada item, cujos resultados estão dispostos na Tabela 11.

Para obter as métricas de erros finais do modelo, calculou-se a média de cada uma das métricas avaliadas e os resultados consolidados estão detalhados na Tabela 12.

Tabela 11 - Métricas de erro por item do modelo GRU

Item	MAE	RMSE	MAPE (%)
Item 1	2,404	2,933	1,210
Item 2	2,017	2,359	2,469
Item 3	2,135	2,299	2,039
Item 4	1,103	1,240	0,927
Item 5	2,777	3,354	5,131
Item 6	6,820	7,688	4,971
Item 7	2,747	3,280	1,514
Item 8	1,542	1,910	0,976
Item 9	1,888	2,329	0,846
Item 10	2,342	2,863	1,899

Fonte: Autoria própria.

Tabela 12 - Métricas de erro finais do modelo GRU

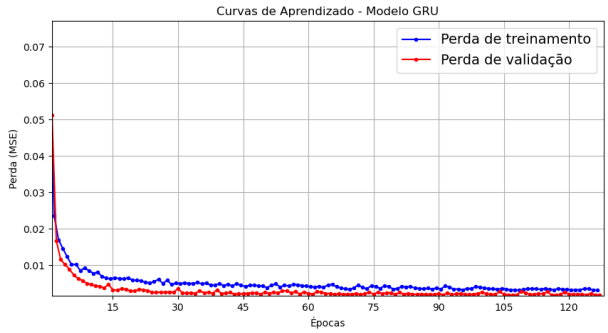
Modelo	MAE	RMSE	MAPE (%)
GRU	2,414	2,897	2,104

Fonte: Autoria própria.

Este modelo apresentou um desempenho quantitativo próximo ao do modelo LSTM. Analisando as métricas para cada item, o MAPE obtido para 6 dos 10 itens ficou abaixo de 2%, sendo que 3 desses itens obtiveram MAPE abaixo de 1%. O maior valor dessa métrica foi observado para o item 5, com um percentual de 5,13%. Os resultados deste modelo apresentaram um MAE de 2,41, um RMSE de 2,90 e um MAPE de 2,10%, sugerindo novamente uma baixa discrepância entre previsões e valores reais. O modelo apresentou um bom desempenho, indicando o segundo menor MAPE dentre todos os modelos.

O gráfico de curvas de aprendizado, contendo a curva de perda de treino e a de perda de validação ao longo das épocas foi gerada e por ser vista na Figura 13.

Figura 13 - Curvas de aprendizado do modelo GRU



Fonte: Autoria própria.

A curva de aprendizado do modelo GRU demonstrou um aprendizado bastante rápido, com ambas as perdas convergindo para valores consideravelmente baixos já entre as primeiras 10 épocas. Apesar disso, as perdas exibiram leves oscilações, indicando sobreajuste, mas de forma sutil e controlada. Da mesma forma que o aprendizado do LSTM, o modelo GRU apresentou efeito da regularização, cuja perda de treinamento se manteve um pouco maior que a curva de validação, mas se aproximaram gradativamente com o passar das épocas, o que provavelmente justifica o sobreajuste controlado. O EarlyStopping estende o treinamento por um número maior de épocas, encerrando-o na época 127, buscando assim maior estabilidade entre as perdas de treinamento e validação.

Em termos gerais, o modelo apresentou métricas de erro boas e um aprendizado rápido. Isso sugere que a arquitetura bidirecional combinada com o *recurrent_dropout* foi eficaz em promover uma boa generalização nas fases iniciais do treinamento, controlando bem o sobreajuste. Porém, apesar do seu bom desempenho, o modelo não chegou a superar os resultados do modelo de RNN simples.

4.7 Comparação entre modelos

Os resultados das métricas de erro dos cinco modelos implementados foram consolidados e estão na Tabela 13 para facilitar sua comparação e análise.

Tabela 13 - Métricas de erro finais dos cinco modelos

Modelo	MAE	RMSE	MAPE (%)
RNN Simples	2,157	2,614	1,785
RNN com *Batch Normalization*	3,938	4,914	3,733
RNN com *Layer Normalization*	3,224	3,738	2,833
LSTM	2,826	3,332	2,561
GRU	2,414	2,897	2,104

Fonte: Autoria própria.

A análise dos resultados mostra que o modelo baseado em RNN simples se revela com o melhor desempenho geral. Isto sugere que para as características das séries temporais analisadas e com o volume de dados utilizado, a complexidade inerente de alguns modelos como LSTM e GRU, não resultou em uma maior acurácia preditiva. A simplicidade do modelo RNN simples pode ter contribuído para uma menor inclinação ao sobreajuste quando comparado com os

modelos RNN com normalização ou com mecanismos que lidam melhor com dependências de longo prazo. Essas características permitiu uma generalização mais eficaz para os dados de teste.

Os modelos GRU e LSTM obtiveram desempenhos inferiores, porém mais próximos ao da RNN simples, indicando que essas arquiteturas são capazes de lidar com séries temporais mais longas, além de avaliarem o fluxo de informação de longo prazo. Uma vez que o conjunto de dados de entrada possui 100 amostras de tempo, o desempenho desses modelos pode ser justificado por esse curto período amostral. O uso de técnicas de bidirecionalidade Dropout e recurrent_dropout mostrou ter sido eficaz no controle do sobreajuste desses modelos, evidenciado por suas curvas de aprendizado que mostraram um aprendizado rápido e um sobreajuste mais sutil. Porém, essas técnicas não foram suficientes para superar o desempenho de uma rede neural recorrente simples. Apesar disso, o modelo GRU apresentou um percentual MAPE apenas 0,319 maior em relação à métrica do RNN simples, enquanto o modelo LSTM apresentou um MAPE apenas 0,776 maior.

Os modelos RNN simples com *Batch Normalization* e *Layer Normalization* obtiveram os desempenhos mais baixos. Suas métricas de erro apresentaram os maiores valores e suas curvas de aprendizado mostraram sinais de sobreajuste mais acentuados e até mesmo mais ruidosos. A vantagem teórica da técnica de LN de ser independente do tamanho do lote e potencialmente mais estável para dados sequenciais, não foi suficiente para alcançar melhores resultados, provavelmente devido à simplicidade dos padrões nos dados em que a RNN simples já conseguia capturar ou à necessidade de ajustes ainda mais finos dos hiperparâmetros da *Layer Normalization*. Similarmente, a técnica de BN se mostrou pouca efetiva, possivelmente pelo fato da técnica poder introduzir ruídos e instabilidades caso essas estatísticas de cada lote não sejam consistentemente representativas ao longo da série temporal, ou seja, a média e o desvio padrão podem ter mudado significativamente de um lote da série para outro.

5 CONCLUSÃO

A análise comparativa das cinco arquiteturas de redes neurais recorrentes implementadas neste trabalho revelou que o modelo RNN simples obteve o desempenho preditivo mais acurado para a previsão de consumo de materiais. Com um erro percentual absoluto médio (MAPE) de apenas 1,785% no conjunto de teste, o modelo demonstrou uma notável capacidade de generalização, indicando que suas previsões desviaram, em média, menos de 1,8% do valor real. Este resultado sugere que, para o conjunto de dados em analisados, a simplicidade da RNN simples representou vantagem, sendo suficiente para capturar os padrões temporais relevantes sem ser tão afetado pelo sobreajuste que pode impactar modelos mais parametrizados.

Já o desempenho dos modelos GRU (2,104% de MAPE) e LSTM (2,561% de MAPE) posicionaram-se como as alternativas mais viáveis, embora não tenham superado a RNN simples. A complexidade adicional dessas arquiteturas, embora teoricamente mais poderosas para dependências de longo prazo, não se traduziu em ganhos de desempenho, possivelmente devido ao volume de dados ou à natureza das dependências presentes nas séries.

Por outro lado, os modelos RNN que utilizaram técnicas de normalização apresentaram resultados notavelmente inferiores. O modelo com *Layer Normalization* (MAPE de 2,833%), embora teoricamente mais adequado para RNNs, não conferiu vantagens competitivas neste cenário. O desempenho mais fraco foi o do modelo com *Batch Normalization* (MAPE de 3,733%), cujas curvas de aprendizado revelaram alta instabilidade na perda de validação, um comportamento que pode ser atribuído à inadequação do cálculo de estatísticas por lote para dados de natureza sequencial, reforçando que a escolha da técnica de normalização deve ser criteriosa.

Uma análise detalhada dos resultados por item, revelou que o item 5 consistentemente apresentou as métricas de erro mais elevadas em múltiplas arquiteturas, destacando-se como um caso atípico. Este comportamento indica que sua série temporal possui características particulares — como maior volatilidade, padrões não-lineares complexos ou um nível de ruído mais acentuado — que dificultaram a modelagem e impactaram negativamente o desempenho agregado. Uma abordagem futura poderia envolver a modelagem isolada deste item ou a aplicação de técnicas de pré-processamento mais específicas para ele.

Conclui-se, assim, que a aplicação de algoritmos de aprendizado de máquina para a previsão de demanda de materiais industriais mostra-se uma abordagem promissora e eficaz.

Os resultados demonstram que mesmo arquiteturas mais simples podem gerar previsões de alta acurácia, permitindo à empresa ajustar com maior precisão parâmetros logísticos críticos, como o estoque de segurança e o ponto de encomenda. Deve-se ressaltar, contudo, que a limitada quantidade de amostras históricas disponíveis representa uma limitação do estudo.

Trabalhos Futuros

- Realizar a otimização dos hiperparâmetros dos modelos, com especial atenção ao refinamento da estratégia de *early stopping*, considerando que algumas curvas de aprendizado apresentaram oscilações na perda de validação antes da interrupção do treinamento.
- Incorporar variáveis temporais explícitas (como mês, dia da semana e feriados) e variáveis externas (como dados de produção e indicadores econômicos), a fim de enriquecer o contexto informacional dos modelos.
- Explorar o uso de redes neurais convolucionais (CNN) para o pré-processamento das séries temporais, visando à extração de padrões locais relevantes antes da modelagem com RNNs.
- Avaliar a estacionariedade das séries temporais e aplicar técnicas de diferenciação, quando necessário, como forma de simplificar a tarefa de modelagem e melhorar o desempenho das redes neurais.

REFERÊNCIAS

ACADEMY, D. S. **Análise e Modelagem de Séries Temporais**: Técnicas Estatísticas, IA e Suas Aplicações. Data Science Academy, 2023. Disponivel em: <https://blog.dsacademy.com.br/analise-e-modelagem-de-series-temporais-tecnicas-estatisticas-ia-e-suas-aplicacoes/>. Acesso em: 24 maio 2025.

AMAZON. **O que é uma previsão?** Amazon, 2023a. Disponivel em: <https://aws.amazon.com/pt/what-is/forecast/>. Acesso em: 20 maio 2025.

AMAZON. **O que é o aprendizado profundo?** Amazon, 2023b. Disponivel em: <https://aws.amazon.com/pt/what-is/deep-learning/>. Acesso em: 24 maio 2025.

BROWNLEE, J. **How to use Learning Curves to Diagnose Machine Learning Model Performance**. Machine Learning Mastery, 2019. Disponivel em: <https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/>. Acesso em: 25 maio 2025.

BRUCE, P.; BRUCE, A. **Estatística Prática para Cientistas de Dados**. 1. ed. Rio de Janeiro: Alta Books, 2019.

CLAUSE FÁTIMA DE BRUM PIANA, A. D. A. M. L. P. R. S. **Estatística Básica**. Universidade Federal de Pelotas. Pelotas. 2013.

COSTA, J. C. D. **Planejamento, programação e controle de produção**. 1. ed. Londrina: Editora e Distribuidora Educacional S.A., 2016.

GÉRON, A. **Mãos à Obra**: Aprendizado de Máquina com Scikit-Learn, Keras & TensorFlow. 2. ed. Rio de Janeiro: Alta Books, 2021.

GRAZIANI, Á. P. **Planejamento, Programação e Controle da Produção**. 1. ed. Palhoça: UnisulVirtual, 2012.

HARRISON, M. **Machine Learning**: Guia de Referência Rápida. 1. ed. São Paulo: Novatec, 2020.

HYNDMAN, R. J.; ATHANASOPOULOS, G. **Forecasting Principles and Practice**. 3. ed. Melbourne: OTexts, 2021.

IBM. **Machine Learning e Ciência de Dados com IBM Watson**. IBM, 2015. Disponivel em: <https://www.ibm.com/br-pt/analytics/machine-learning>. Acesso em: 08 abr. 2025.

LUSTOSA, L. et al. **Planejamento e Controle da Produção**. 4. ed. Rio de Janeiro: Elsevier, 2008.

MARTINS, P. G.; LAUGENI, F. P. **Administração da Produção**. 3. ed. São Paulo: Saraiva, 2007.

MATPLOTLIB, E. **Matplotlib**: Visualização com Python. Matplotlib, 2025. Disponivel em: <https://matplotlib.org/>. Acesso em: 22 mar. 2025.

MEDRI, W. **Análise Exploratória de Dados**. Universidade Estadual de Londrina. Londrina, . 2011.

NIELSEN, A. **Análise Prática de Séries Temporais**: Predição com Estatística e Aprendizado de Máquina. 1. ed. Rio de Janeiro: Alta Books, 2021.

NUMFOCUS. **About pandas**. Pandas, 2025. Disponivel em: <https://pandas.pydata.org/about/>. Acesso em: 22 mar. 2025.

NUMPY, E. **NumPy**. Numpy, 2025. Disponivel em: <https://numpy.org/>. Acesso em: 22 mar. 2025.

PENTEADO, K. **Métricas de avaliação para séries temporais**. Alura, 2021. Disponivel em: <https://www.alura.com.br/artigos/metricas-de-avaliacao-para-series-temporais>. Acesso em: 16 maio 2025.

SCIKIT-LEARN, E. **scikit-learn Machine Learning in Python**. Scikit-learn, 2025. Disponivel em: <https://scikit-learn.org/stable/>. Acesso em: 22 mar. 2025.

SOUZA, V. N. D. **Redes Neurais Artificiais Aplicadas a Previsão de Curto Prazo de Séries Temporais Financeiras**. 2018. Trabalho de Conclusão de Curso (Bacharelado em Engenharia Eletrica) - Escola de Engenharia, Universidade de São Carlos, São Paulo, 2018.

SUNIGA, A. **Conjuntos de treino, validação e teste em Machine Learning**. Medium, 2020. Disponivel em: <https://medium.com/@abnersuniga7/conjuntos-de-treino-teste-e-valida%C3%A7%C3%A3o-em-machine-learning-fast-ai-5da612dcb0ed>. Acesso em: 25 maio 2025.

TENSORFLOW, E. **An end-to-end platform for machine learning**. An end-to-end platform for machine learning, 2025. Disponivel em: <https://www.tensorflow.org/>. Acesso em: 23 mar. 2025.

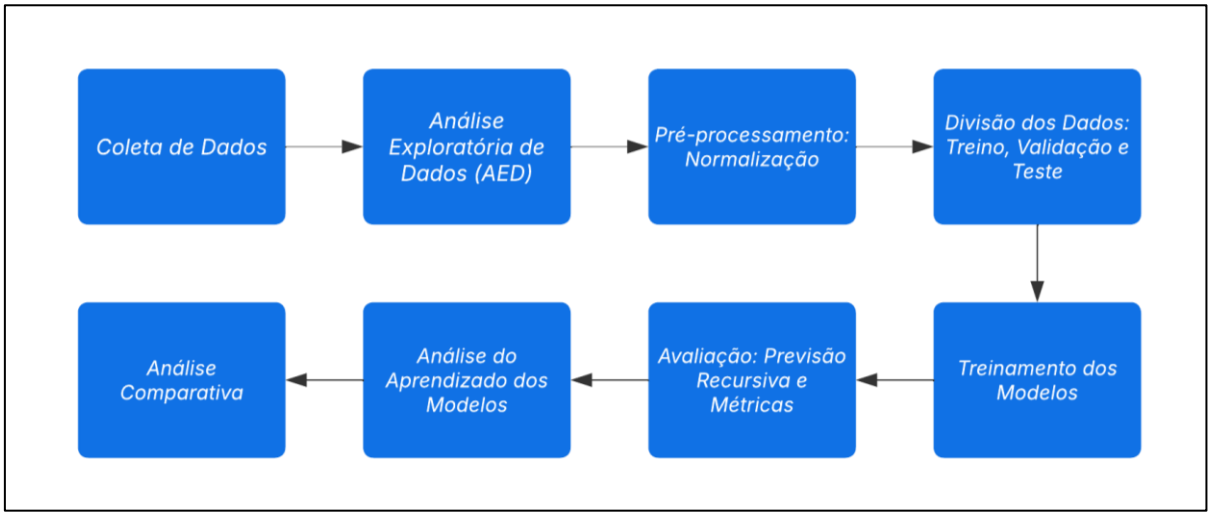
TOTVS. **Ressuprimento por Ponto de Encomenda**. Totvs, 2018. Disponivel em: <https://tdn.engpro.totvs.com.br/display/public/LDT/Ressuprimento+por+Ponto+de+Encomenda>. Acesso em: 7 jun. 2025.

TOTVS. **Descubra os benefícios do PPCP para a indústria**. Totvs, 2020. Disponivel em: <https://www.totvs.com/blog/gestao-industrial/ppcp/>. Acesso em: 7 jun. 2025.

TUBINO, D. F. **Planejamento e controle da produção**. 1. ed. São Paulo: Atlas Editora, 2007.

APÊNDICE A – FLUXOGRAMA DA METODOLOGIA DE DESENVOLVIMENTO UTILIZADA NO PROJETO

Figura 14 - Fluxograma da metodologia utilizada no projeto



Fonte: Autoria própria.