THIAGO MANTUANI DE SOUZA

PREVISÃO DE PREÇOS E DEMANDAS DE PRODUTOS DO VAREJO UTILIZANDO TÉCNICAS DE APRENDIZADO DE MÁQUINA

LAVRAS - MG

2021

THIAGO MANTUANI DE SOUZA

PREVISÃO DE PREÇOS E DEMANDAS DE PRODUTOS DO VAREJO UTILIZANDO TÉCNICAS DE APRENDIZADO DE MÁQUINA

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Engenharia de Sistemas e Automação, para a obtenção do título de Mestre.

Dr. Wilian Soares Lacerda
Orientador

LAVRAS – MG

2021

Ficha catalográfica elaborada pelo Sistema de Geração de Ficha Catalográfica da Biblioteca Universitária da UFLA, com dados informados pelo(a) próprio(a) autor(a).

Souza, Thiago Mantuani de.
Previsão de preços e demandas de produtos do varejo utilizando técnicas de aprendizado de máquina / Thiago Mantuani de Souza. - 2021.
106 p. : il.

Orientador(a): Wilian Soares Lacerda.

Dissertação (mestrado acadêmico) - Universidade Federal de Lavras, 2021.
Bibliografia.

1. Aprendizado de máquina. 2. Demanda. 3. Varejo. I. Lacerda, Wilian Soares. II. Título.

THIAGO MANTUANI DE SOUZA

PREVISÃO DE PREÇOS E DEMANDAS DE PRODUTOS DO VAREJO UTILIZANDO TÉCNICAS DE APRENDIZADO DE MÁQUINA

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Engenharia de Sistemas e Automação, para a obtenção do título de Mestre.

APROVADA em 28 de junho de 2021.

Dr. José Willer do Prado	UFLA
Dr. Marcelo Azevedo Costa	UFMG
Dr. Wilian Soares Lacerda	UFLA

Wilian Soares Lacerda
Dr. Wilian Soares Lacerda
Orientador

LAVRAS – MG

2021

Dedico este trabalho à minha família.

AGRADECIMENTOS

Agradeço primeiramente a Deus, por colocar pessoas especiais ao meu lado e por fortalecer minhas forças a todo tempo.

Aos meus pais Dolor e Maria por me apoiarem nos momentos mais difíceis.

À mãe da minha filha, pela paciência, apoio e disposição para me ajudar.

A minha filha, que em meus momentos mais difíceis me alegrou.

Ao meu orientador, Prof.º Dr. Wilian Soares Lacerda, pela disponibilidade, compreensão e ensinamentos.

Aos membros da minha banca, por aceitarem contribuir com meu trabalho.

À Universidade Federal de Lavras, e ao Programa de Pós-Graduação em Engenharia de Sistemas e Automação, pela estrutura e oportunidade para realização do mestrado.

A todos os amigos e colegas que de alguma forma contribuíram direta ou indiretamente na escrita deste trabalho.

MUITO OBRIGADO!

“Os erros causados por dados inadequados são muito menores do que aqueles devido à falta total de dados.” (Charles Babbage)

RESUMO

O sucesso das empresas do varejo depende de alguns fatores que auxiliam na tomada de decisão. Um desses fatores está relacionado à estocagem e disponibilidade dos produtos, para assim atender a demanda dos clientes. Os preços também são um desses fatores, pois, a partir deles, os clientes tomarão a decisão de compra dos produtos. Desta forma, o objetivo deste trabalho foi aplicar técnicas de aprendizado de máquina (AM) para prever as demandas e os preços de alguns produtos do varejo. Para o treinamento do sistema de AM, escolheu-se uma série de vendas de alguns produtos no varejo que compreende o período de abril/2015 a dezembro/2019 na cidade de Cambuí/MG. As técnicas de AM aplicadas e comparadas foram: Regressão Linear, Rede Neural Artificial *Multilayer Perceptron*, Rede Neural Recorrente *Long Short Term Memory*, Máquinas de Vetores de Suporte, K Vizinhos mais Próximos e Floresta Aleatória (FA). Os resultados das previsões de demandas e preços foram obtidos através das vendas diárias e avaliados através das métricas da raiz quadrada do erro médio (RMSE), raiz do erro logarítmico quadrático médio (RMSLE), erro médio absoluto (MAE) e coeficiente de determinação (R^2). Após a execução dos modelos de AM referentes a treze períodos distintos obteve-se o RMSE, RMSLE, MAE e R^2 de cada um desses períodos. Posteriormente, aplicou-se o teste não paramétrico de *Friedman* para verificar se havia diferença estatística entre as médias e o teste de *Nemenyi* para identificar quais modelos eram diferentes. O modelo de FA apresentou as melhores previsões para os preços e demandas de produtos do varejo. Neste caso, os valores calculados para as métricas RMSE, RMSLE, MAE e R^2, através da FA para a previsão de preços foram próximos a 0,07, 0,03, 0,11 centavos e 0,99 respectivamente. Na previsão de demanda quando foi aplicado o algoritmo FA, o valor calculado para o RMSE foi aproximadamente 55,6, enquanto o valor do RMSLE calculado foi de 0,63 e o MAE foi próximo a 4 unidades de produtos. Por fim, o valor encontrado para o R^2 foi de 0,57. Sendo assim, a FA provou ser um método eficiente para previsão de preços e demandas dos produtos do varejo abordados neste trabalho.

Palavras-chave: Aprendizado de Máquina. Demanda. Floresta Aleatória. Varejo. K Vizinhos mais próximos. Máquinas de Vetores de Suporte. Preços. Rede Neural Artificial.

ABSTRACT

The success of retail companies depends on some factors that help in decision making. One of these factors is related to the storage and availability of products, in order to meet customer demand. Prices are also one of these factors because, based on them, customers will make the decision to purchase the products. Thus, the objective of this work was to apply machine learning (ML) techniques to predict the demands and prices of some retail products. For the ML system training, a series of retail sales of some products was chosen, covering the period from April/2015 to December/2019 in the city of Cambuí/MG. The ML techniques applied and compared were: Linear Regression, Multilayer Perceptron Artificial Neural Network, Long Short Term Memory Recurrent Neural Network, Support Vector Machines, K Nearest Neighbors and Random Forest (RF). The results of demand and price forecasts were obtained through daily sales and evaluated through the metrics of the root mean square error (RMSE), root mean square logarithmic error (RMSLE), mean absolute error (MAE) and coefficient of determination (R^2). After the execution of the ML models referring to thirteen different periods, the RMSE, RMSLE, MAE and R^2 of each of these periods were obtained. Subsequently, Friedman's non-parametric test was applied to verify whether there was a statistical difference between the means and the Nemenyi test to identify which models were different. The RF model provided the best predictions for retail product prices and demands. In this case, the values calculated for the RMSE, RMSLE, MAE and R^2 metrics, through the RF for price forecasting, were close to 0.07, 0.03, 0.11 cents and 0.99 respectively. In the demand forecast when the RF algorithm was applied, the calculated value for the RMSE was approximately 55.6, while the calculated RMSLE value was 0.63 and the MAE was close to 4 product units. Finally, the value found for R^2 was 0.57. Thus, RF proved to be an efficient method for forecasting prices and demand for retail products covered in this work.

Keywords: Artificial Neural Network. Demand. K Nearest neighbors. Machine Learning. Prices. Random Forest. Retail. Support Vector Machines.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O varejo é uma modalidade de venda de produtos e serviços realizado diretamente para o consumidor final. Nele, o consumidor se dirige há um ponto de compra que pode ser físico ou virtual, no qual ele demanda por produtos ou serviços de sua escolha. Nesse sentido, os varejistas buscam a cada dia expandir suas tecnologias para atrair a atenção dos consumidores, além de fornecer um melhor atendimento (MATTAR, 2011; PARENTI; BARKI, 2014).

As vendas são as principais atividades desempenhadas pelos varejistas, e as previsões podem fornecer aos gestores informações valiosas para o planejamento operacional e financeiro, impactando diretamente em diversas funções da empresa, como na projeção de custos, lucros e no planejamento de compras (GEURTS; KELLY, 1986; RAMOS; SANTOS; REBELO, 2015).

As previsões de preços e demandas podem ser realizadas por métodos quantitativos, os quais são aplicados no cotidiano de forma recorrente. Um exemplo muito utilizado são os métodos lineares como técnicas estatísticas de séries temporais. Métodos não lineares utilizando técnicas de aprendizado de máquinas (AM) tais como: Redes Neurais Artificiais (RNA’s), Máquinas de Vetores de Suporte (SVM) e K vizinhos mais próximos (KNN) têm sido utilizados com bastante sucesso (FRANK et al., 2003; YUE et al., 2007).

O AM é um componente da Inteligência Artificial (IA) responsável por criar modelos computacionais que aprendem através da experiência (FACELI et al., 2011). A aplicação desses referidos modelos, pode ser uma boa opção não linear para técnicas como a regressão ou modelos ARIMA, análise de discriminante, reconhecimento de padrões e séries temporais, entre outros.

Nos modelos supracitados, as abordagens com séries temporais possibilitam a organização de informações quantitativas num espaço de tempo, porém, as séries de tempo se constituem de dados históricos ocorridos ao longo do tempo. No varejo, as análises das séries temporais de vendas possibilitam previsões sobre o futuro, através do comportamento das vendas realizadas no passado. Desta forma, a contribuição da aplicação de técnicas de aprendizado de máquina nas séries temporais para este tipo de mercado, inclui diminuição de erros com relação à estocagem das mercadorias e ao planejamento adequado para que não haja excesso ou falta de produtos (PARENTE; BARKI, 2014).

1.1 Objetivo Geral

Objetiva-se com este trabalho investigar e testar técnicas de aprendizado de máquinas para previsão de preços e demandas de 20 produtos do varejo, através de dados históricos de vendas de uma rede varejista da cidade de Cambuí/MG.

1.2 Objetivos Específicos

Este trabalho também almeja os seguintes objetivos específicos:

- Seleção das melhores características de dados para a previsão de preço e demanda de produtos do varejo.
- Aplicação dos modelos de Regressão linear, Redes Neurais Artificiais, Máquina de Vetores de Suporte, Florestas Aleatórias e KNN na previsão de preços e demandas de produtos.
- Comparação entre as técnicas utilizadas nas previsões para determinar a melhor.

1.3 Justificativa

As empresas do varejo nos dias atuais se veem na necessidade de melhorar o planejamento e a tomada de decisões, devido a um ambiente competitivo no qual estão inseridas. As previsões de demanda e preços podem auxiliar os varejistas a terem resultados maiores e melhores, tanto no quesito competitividade quanto no relacionamento com clientes (MOON; MENTZER; SMITH, 2003).

Sendo assim, é importante ressaltar a especificidade do varejo no segmento de supermercados, pois o mesmo trabalha com diversos produtos de diferentes categorias que possuem diferentes frequências de vendas demandadas pelos consumidores. O processo de previsão de vendas e preços para os produtos implica em características como: o horizonte de previsão, de curto, médio ou longo prazo e o impacto da sazonalidade.

Desta forma, este trabalho busca encontrar modelos de previsão para produtos, no intuito de garantir um menor erro, que não ocasionem falta ou excesso de estoque, e auxiliem na tomada de decisões.

1.4 Estrutura do trabalho

O presente trabalho está dividido em 6 capítulos, sendo organizados da seguinte forma. O Capítulo 1 apresenta a parte introdutória, sendo apresentados os objetivos e a justificativa. O Capítulo 2 apresenta o estado da arte em: técnicas de aprendizado de máquinas, vendas no varejo, índice nacional de preços ao consumidor amplo, séries temporais, seleção de características e métricas de avaliação de modelos. O Capítulo 3 apresenta a metodologia utilizada na dissertação, como coleta e entendimento dos dados, análises exploratórias, ferramentas utilizadas, definição e validação dos modelos de aprendizado de máquina. O Capítulo 4 demonstra os resultados encontrados para cada uma das séries temporais definidas no trabalho. No Capítulo 5 são apresentadas as conclusões e propostas para trabalhos futuros. Por fim são apresentadas as Referências Bibliográficas e apêndices (A e B), contendo os códigos-fonte e figuras com tendências dos produtos.

2 REFERENCIAL TEÓRICO

Neste capítulo, abordar-se-á o levantamento bibliográfico com as principais informações e conceitos que fundamentaram esta dissertação.

2.1 Vendas de produtos no Varejo

O varejo é uma modalidade comercial, que se estrutura a partir da busca até a entrega do produto ou serviço ao consumidor final. Os principais varejistas são lojas de conveniência, supermercados, *showroom* de vendas por catálogo, lojas de departamentos, dentre outros (KOTLER; KELLER, 2006; SANTOS; COSTA, 1997).

A história do varejo no Brasil e em outras civilizações, iniciou-se pelo processo de troca, também denominado escambo, que foi considerado o primeiro sistema de comércio nas sociedades. No período colonial do Brasil, devido às plantações de monoculturas como o café e a cana de açúcar, surgem os primeiros armazéns (VAROTTO, 2006). Segundo Mattar (2011), desde então as experiências internacionais positivas contribuíram para moldar e modernizar o varejo.

Uma métrica para se verificar a relevância do varejo é verificar a sua participação no Produto Interno Bruto (PIB) que mede os bons resultados obtidos na economia do país. No Brasil, em 2018, a participação do comércio varejista apresentou montante de R$ 6,6 trilhões de acordo com a Sociedade Brasileira de Varejo e Consumo (2018), sendo que este valor equivale a 63,4% do PIB nacional.

As vendas do varejo representam o total de faturamento obtido através do cliente, mas esse valor pode ou não conter deduções, devoluções e abatimentos sobre as vendas (IUDÍCIBUS; MARION, 2016). Segundo Parente e Barki (2014), as vendas cujos valores sofrem deduções, devoluções e abatimentos são denominadas vendas líquidas ou receitas líquidas. Entretanto, as vendas de produtos livres de deduções, devoluções e abatimentos, que possuem incidência de impostos sobre elas denominam-se receita bruta (GRECO; AREND, 2013; MARION, 2015).

As empresas e organizações comerciais possuem um estoque de mercadorias, o qual se constituí em um elemento de custo, que geralmente são os produtos obtidos para revenda. O custo pode ser definido como o valor pago por algo e possui visões diferentes para o vendedor e o comprador. Assim, o produto a ser vendido representa o preço de venda para o vendedor (quem vende) e o custo unitário para o comprador (quem compra) (PADOVEZE, 2013). Desta forma, com a saída da mercadoria, e com todo o estoque vendido, o valor de aquisição

dos produtos entregues aos clientes torna-se uma despesa para a empresa, conhecido como Custo da Mercadoria Vendida (CMV) (PADOVEZE, 2013).

De acordo com Parenti e Barki (2014), a geração e otimização dos lucros é um dos principais objetivos do comércio varejista. O Lucro Bruto é um dos indicadores de desempenho da empresa, ele é resultado da diferença entre as receitas líquidas e o custo das mercadorias vendidas, ou seja, subtrai-se da receita líquida o CMV, e desconsidera as despesas financeiras, administrativas e de vendas (IUDÍCIBUS; MARION, 2016). Além disso, à medida que o lucro bruto aumenta, maior é a possibilidade de renda dos administradores, proprietários da empresa, dentre outros interessados no negócio (MARION, 2015).

2.1.1 Previsão de demanda

A previsão de demanda é um fator importante para o planejamento estratégico de vendas de uma empresa, pois possui um papel fundamental no gerenciamento de estoques, uma vez que o erro de previsões pode causar faltas e/ou excessos de estoques de produtos (PARENTI; BARKI, 2014).

As técnicas de previsões podem ser classificadas em dois tipos: técnicas quantitativa e qualitativa, e elas podem ser de curto, médio e longo prazo. Os métodos quantitativos são baseados em séries de dados históricos, através de séries temporais e regressão, já as qualitativas são baseadas em opiniões de especialistas e pesquisas de mercado (CORRÊA; CORRÊA, 2019; KRAJEWSKI; RITZMAN; MALHOTRA, 2009).

Os métodos mais utilizados são os quantitativos e podem ser divididos em duas categorias: técnicas fundamentadas em séries temporais e em correlações. A primeira abordagem propõe descobrir padrões nas informações passadas disponíveis, através dos componentes de uma série temporal. Por outro lado, técnicas baseadas em correlações associam os dados do produto com outras variáveis relacionadas à demanda (KRAJEWSKI; RITZMAN; MALHOTRA, 2009; SILVA; OLVEIRA, 2012; TUBINO, 2007).

Um modelo de previsão é dividido em cinco etapas: inicialmente, define-se o objetivo do modelo; em seguida inicia-se a coleta de dados históricos dos produtos; seleciona-se a técnica a ser utilizada para previsão; realiza as previsões de curto, médio ou longo prazo, e por fim, obtêm-se o resultado das previsões e o monitoramento do modelo (TUBINO, 2007).

2.1.2 Demanda de mercado

A demanda é definida como a quantidade de determinado produto ou serviço que os consumidores visam comprar em determinado período. A procura por essa aquisição de bens e produtos está relacionada com as variáveis que influenciam diretamente no desejo do consumidor tais como: preços, renda e interesses do consumidor (VASCONCELOS; GARCIA, 2019).

A relação entre a quantidade demandada pelo consumidor e o preço é conhecido como lei da demanda. Essa relação é inversamente proporcional, ou seja, quanto maior o preço do produto, menos consumidores procurarão esse produto (SILVA; SILVA, 2018; VASCONCELOS; GARCIA, 2019). De acordo com Silva e Silva (2018), a definição dos preços depende da interatividade da oferta e demanda, e muitas vezes o preço depende também da estrutura de mercado em que os bens e produtos se enquadram.

2.2 Técnicas de aprendizado de máquina

2.2.1 Regressão Linear

O modelo de regressão linear é utilizado para realizar previsões ou medir a relação entre uma ou mais variáveis independentes/explicativas e a variável dependente, geralmente denotadas por x e y respectivamente. Ela pode ser classificada como simples e múltipla (YAN; SU, 2009).

A regressão linear simples pode ser descrita conforme Equação 2.1, em que y é a variável dependente, β_0 o intercepto, x a variável independente, β_1 o coeficiente angular e ε o termo do erro aleatório (MONTGOMERY; PECK; VINING, 2012).

$$y = \beta_0 + \beta_1 x + \varepsilon \tag{2.1}$$

Para encontrar os coeficientes β_0 e β_1, utiliza-se o método dos mínimos quadrados. A Equação 2.2 apresenta o cálculo para encontrar o coeficiente β_1, em que n representa o número de amostras.

$$\beta_1 = \frac{n\,(\sum xy) - (\sum x)\,(\sum y)}{n\,(\sum x^2) - (\sum x)^2} \tag{2.2}$$

O coeficiente β_0 pode ser calculado conforme Equação 2.3, em que $\bar{y}$ refere-se à média dos valores de *y*, e $\bar{x}$ a média dos valores de *x*. O cálculo dos coeficientes tem como objetivo minimizar a soma dos quadrados dos desvios (LAPPONI, 2005).

$$\beta_0 = \bar{y} - \beta_1 \bar{x} \tag{2.3}$$

A regressão linear múltipla é aplicada quando há mais de uma variável explicativa (*x*) no modelo, e pode ser descrita conforme Equação 2.4. Em que β_0, β_1 e β_n são os coeficientes de regressão e x_1, x_2 e x_n, são as variáveis explicativas e ε o erro aleatório (MONTGOMERY; PECK; VINING, 2012).

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \varepsilon \tag{2.4}$$

O modelo de regressão linear múltipla pode ser reescrito em forma matricial, conforme Equação 2.5 (MORAIS, 2010). Em que Y é o vetor das observações, X é uma matriz das variáveis independentes, β um vetor dos coeficientes e ε o vetor dos erros.

$$Y = X\beta + \varepsilon \tag{2.5}$$

Assim como a regressão linear simples, a regressão múltipla também utiliza o método dos mínimos quadrados para estimar os coeficientes de regressão (Equação 2.6).

$$\hat{\beta} = (X^T X)^{-1} X^T Y \tag{2.6}$$

2.2.2 Redes Neurais Artificiais

As Redes Neurais Artificiais (RNA's) são modelos computacionais que simulam o aprendizado e possuem a capacidade de generalização. Foram desenvolvidas baseadas no cérebro humano e elas podem ser utilizadas em tarefas de regressão, classificação, reconhecimento de padrões e séries temporais, dentre outros (BRAGA; CARVALHO; LUDERMIR, 2007).

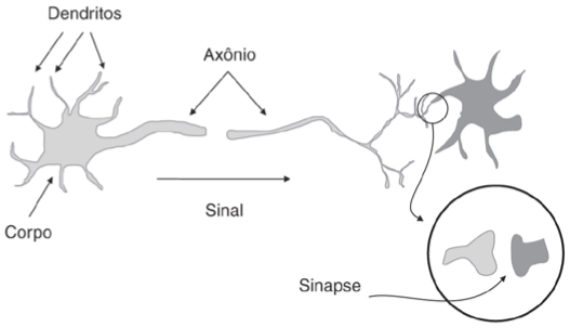
2.2.2.1 Neurônio Artificial

As Redes Neurais Artificiais (RNA's) são formadas por unidades de processamento simples e interconectadas, denominadas neurônios. Através desse modelo computacional

análogo ao neurônio do Sistema Nervoso Central (SNC) é possível o reconhecimento de padrões, a capacidade de aprendizado e a generalização do conhecimento aprendido (HAYKIN, 2001).

No modelo biológico, as células neurais que compõe o SNC são formadas por três partes fundamentais (FIGURA 1): os dendritos que recebem os estímulos transmitidos por outros neurônios; o corpo celular que é responsável por processar os estímulos recebidos; e o axônio que transmite os estímulos para a célula seguinte. Através das sinapses, conduzidas por essas estruturas celulares, há uma resposta que pode ser interpretada como um resultado a um estímulo anterior (FERNEDA, 2006; GORDAZI et al., 2014).

Figura 1 – Neurônio biológico.

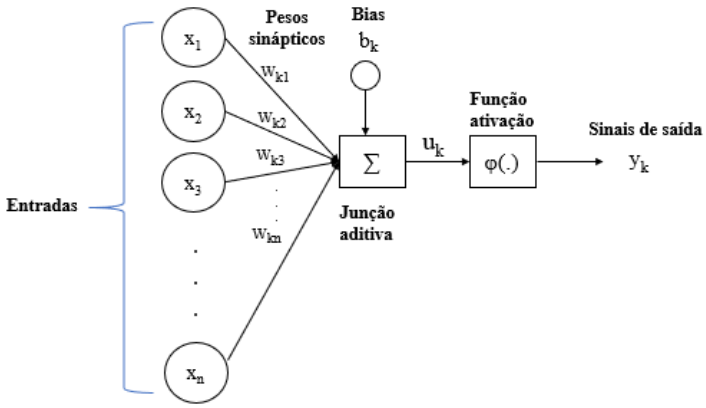


Fonte: Faceli et al. (2011).

O desenvolvimento de um modelo computacional semelhante ao modelo do neurônio biológico para integrar a área de inteligência artificial, foi descrita inicialmente por McCulloch e Pitts (1943), que propuseram o modelo do neurônio artificial. O modelo de McCulloch e Pitts consiste em um neurônio que funciona como um circuito binário e executa funções lógicas simples. Posteriormente, Rosenblatt (1958) melhorou o modelo até então conhecido, e desenvolveu o *Perceptron* de camada única para lidar com o problema de reconhecimento de padrões, e assim surge um novo modelo de neurônio artificial.

O modelo do neurônio artificial do tipo *Perceptron* (FIGURA 2) possui entradas que correspondem aos dendritos, uma junção aditiva que corresponde ao corpo celular e uma saída que é análoga aos axônios. As duas estruturas iniciais, são responsáveis por coletar informações vindas de outros neurônios, e a saída transmite o sinal para outro neurônio ou para um fim. O modelo possui uma função de ativação que tem como objetivo limitar a saída e um *bias (*b_k*)* que tem a funcionalidade de aumentar ou diminuir a entrada líquida da função de ativação (HAYKIN, 2001).

Figura 2 – Neurônio artificial *Perceptron*.



Fonte: Adaptado de Haykin (2001).

O neurônio artificial *Perceptron* pode ser descrito conforme a Equação 2.7, e seu valor de saída (y_k) pode ser calculado mediante aplicação de uma função de ativação(φ) (Equação 2.8) (FACELI et al., 2011).

$$u_k = \sum_{j=1}^{n} w_{kj} x_j + b_k \tag{2.7}$$

$$y_k = \varphi(u_k) \tag{2.8}$$

O treinamento do neurônio artificial *Perceptron* baseia-se no aprendizado descrito por Hebb (1949), que tem o objetivo de ajustar e encontrar os pesos (*w*) de tal forma que a saída da rede se aproxime da saída desejada. No treinamento do *Perceptron*, deve haver uma constante (α), que define a taxa de aprendizado, e fornecidos os padrões de entrada (*x*) e a saída desejada (*yd*), conforme Haykin (2001). Para cada amostra apresentada a rede, a saída y_k é calculada. Através do valor obtido para y_k, é possível determinar o erro (e_k) conforme a equação (2.9).

$$e_k = (yd_k - y_k) \tag{2.9}$$

Os pesos são atualizados conforme a Equação 2.10 e a taxa de aprendizado (α) é responsável por determinar a velocidade em que os pesos devem ser atualizados. Segundo Faceli et al. (2011), a taxa de aprendizado com valor alto pode causar grandes oscilações, por outro lado, os valores baixos podem gerar poucas variações nos pesos.

$$w_k(n+1) = w_k(n) + \alpha e_k x_n \tag{2.10}$$

Uma das grandes limitações do treinamento do neurônio artificial *Perceptron* é que o conjunto de dados deve ser linearmente separável, para que, dessa forma, seja possível garantir que os pesos encontrados serão adequados para generalizar novos dados (BISHOP, 1997).

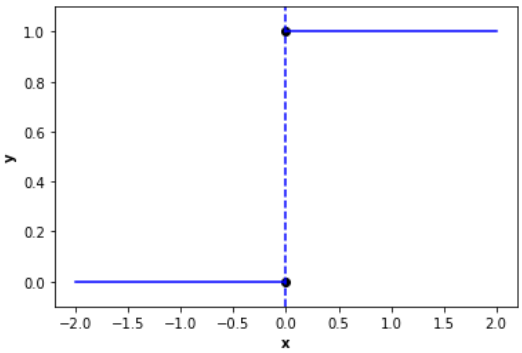
2.2.2.2 Funções de ativação

A função de ativação tem por objetivo gerar a saída do neurônio através dos pesos e das entradas, além de modelar associações não lineares entre os resultados previstos e reais nas RNA's (CHAND; ALAM; TEJASWINI, 2015). Os principais tipos de função de ativação são:

- Função de Limiar (FIGURA 3), em que:

$$\varphi(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{2.11}$$

Figura 3 – Gráfico da função de limiar (Degrau).

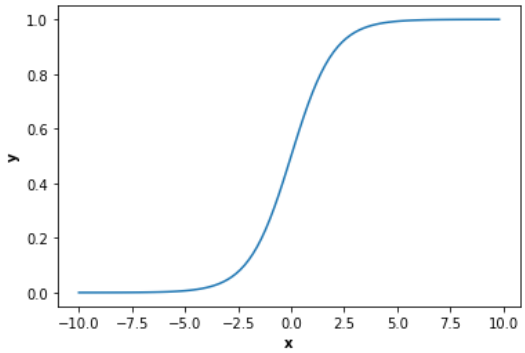


Fonte: Do autor (2019).

- Função sigmoide ou logística (FIGURA 4), em que:

$$\varphi(x) = \frac{1}{1 + e^{(-x)}} \tag{2.12}$$

Figura 4 – Gráfico da função sigmoide ou logística.

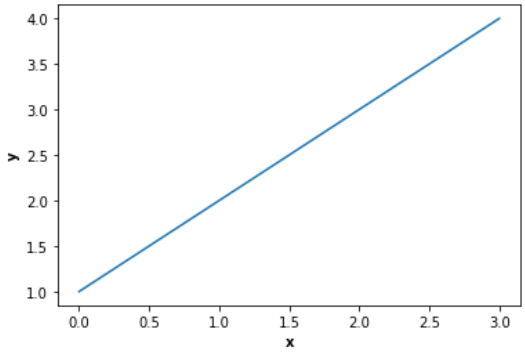


Fonte: Do autor (2019).

- Função linear (FIGURA 5), em que:

$$\varphi(x) = ax + b \tag{2.13}$$

Figura 5 – Gráfico da função linear.

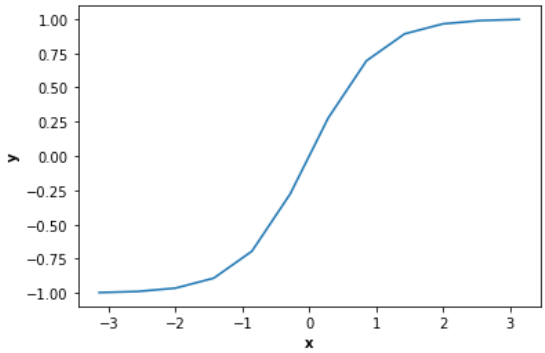


Fonte: Do autor (2019).

- Função tangente hiperbólica (FIGURA 6), em que:

$$\varphi(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.14}$$

Figura 6 – Gráfico da função tangente hiperbólica.

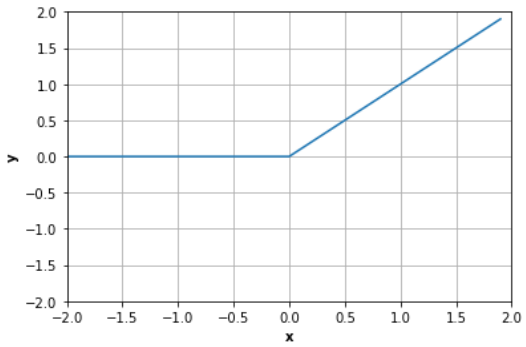


Fonte: Do autor (2019).

- Função de Ativação linear retificada – ReLU (FIGURA 7), em que:

$$\varphi(x) = \max\{0, x\} \tag{2.15}$$

Figura 7 – Gráfico da função ReLU.



Fonte: Do autor (2019).

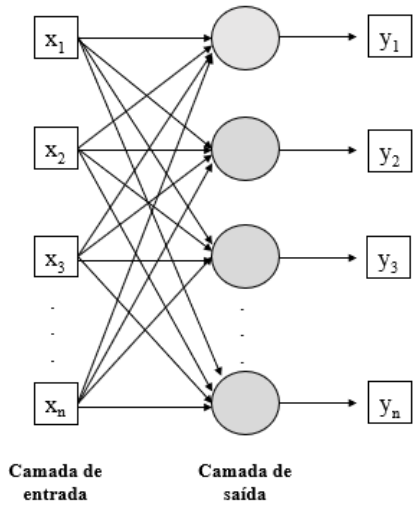
2.2.2.3 Arquitetura da Rede Neural Artificial

Em uma rede neural artificial (RNA), os neurônios estão dispostos em uma ou mais camadas (FACELI et al., 2011). A arquitetura da RNA está relacionada com o algoritmo de aprendizado utilizado para os ajustes dos pesos durante o treinamento, além da quantidade de neurônios e camadas (FAUSETT, 1994). As redes podem ser divididas em *feedforward* de uma única camada e de múltiplas camadas.

Na arquitetura *feedforward* de uma camada a RNA possui apenas uma única camada, que é referente à camada de saída, e a de entrada é utilizada para apresentação dos dados (HAYKIN, 2001). A primeira RNA com apenas uma camada, foi implementada com o neurônio artificial *Perceptron*, capaz de resolver apenas problemas linearmente separáveis (FACELI et al., 2011; MINSKY; PAPERT, 1969) (FIGURA 8).

Um novo modelo de RNA semelhante ao *Perceptron* foi desenvolvido por Widrow e Hoff (1960), conhecido como *Adaline* (*Adaptive Linear Element*), a diferença entre ambos estava na função de ativação, pois, ao invés da função de limiar, neste novo modelo, utilizou-se uma função linear. Em redes de camada única, assim como em *Adaline* e *Perceptron*, os sinais são propagados sempre da camada de entrada para a saída (*feedforward*) (FIGURA 8).

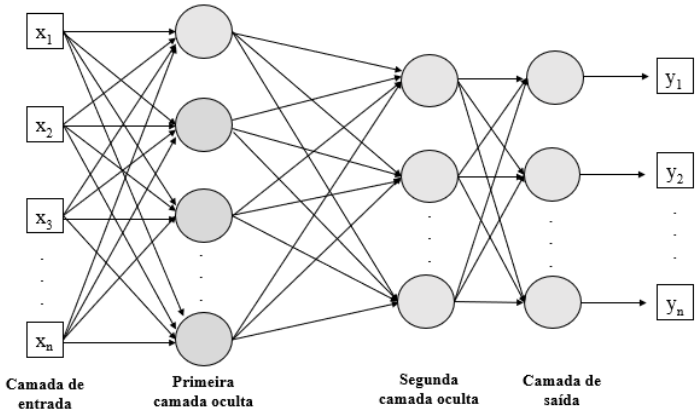
Figura 8 – Rede *feedforward* de uma camada.



Fonte: Adaptado de Braga; Carvalho; Ludemir (2007).

As RNAs de múltiplas camadas distinguem-se das redes de uma única camada devido à presença de uma nova camada, denominada camada escondida, intermediária ou oculta, presente em sua estrutura. A camada escondida fornece à RNA um maior poder computacional na aproximação de funções, entretanto, uma rede com duas camadas escondidas pode implementar qualquer função (BRAGA; CARVALHO; LUDERMIR, 2007; CYBENKO, 1989). A rede de múltipla camada mais popular é a *Multilayer Perceptron* (MLP), que contém uma camada de entrada, uma ou mais camadas escondidas e uma de saída (FIGURA 9).

Figura 9 – Rede *feedforward* de três camadas.



Fonte: Adaptado de Braga; Carvalho; Ludemir (2007).

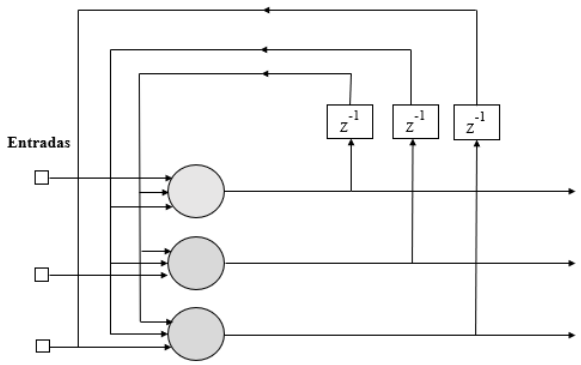
Algumas variáveis são importantes de serem quantificadas para implementar uma RNA, tais como: os números de entradas, camadas escondidas e seus neurônios, além da camada de saída e seus neurônios (SANTOS et al., 2005). Não existe um consenso sobre qual número de neurônios escolher para as camadas escondidas, contudo, se a camada possuir muitos neurônios a rede poderá memorizar os dados ou os padrões, e ser ineficaz para previsões de novos resultados, abordagem conhecida como *overfitting* (ANDREONI; YIP, 2020; JABBAR; KHAN, 2014). A consequência dessa memorização é perder sua capacidade de generalização, mas, se a quantidade for pequena ela não será capaz de identificar e modelar a estrutura dos dados mais complexos, não convergindo durante o treinamento, fenômeno conhecido como *underfitting* (CAUDILL, 1990; MAS; FLORES, 2008; ROSIN; FIERENS, 1995).

2.2.2.4 Redes recorrentes

As redes neurais artificiais recorrentes (*Recurrent Neural Network*) foram introduzidas a partir do modelo proposto por Hopfield (1982), que relatou a utilização de redes simétricas para otimização e apresentou uma rede auto associativa em que as saídas estão ligadas às entradas. Este tipo de rede se diferencia das RNA's *feedforward* por possuírem um ciclo de realimentação que contém ao menos uma interligação realimentada pela saída do neurônio para outros neurônios, fluindo a informação nas duas direções da rede (HAYKIN, 2001). O processo de realimentação faz com que a rede tenha a capacidade de armazenar as informações no tempo de maneira similar a uma memória. O modelo de rede recorrente com uma camada contendo três neurônios com realimentação de saída é apresentado na Figura 10.

Desta forma, os elementos z^{-1} fazem com que a rede tenha um comportamento não linear e dinâmico.

Figura 10 – Rede recorrente.



Fonte: Adaptado de Haykin (2001).

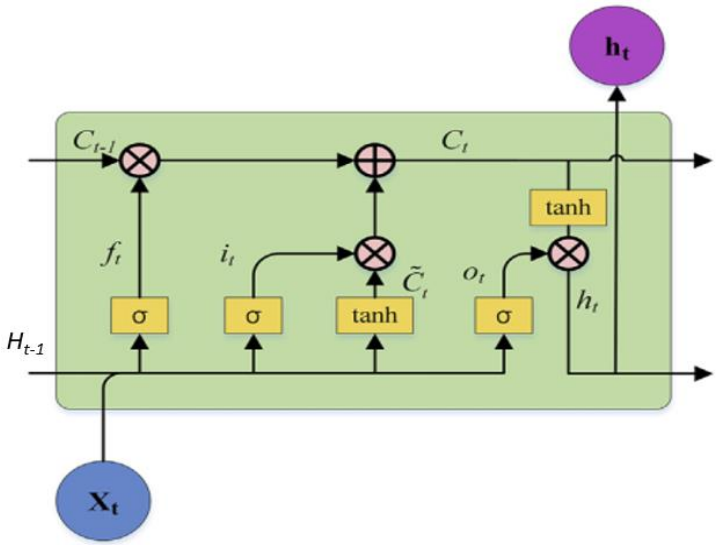
As RNNs foram projetadas para dados sequenciais, que podem ser aplicadas em diversas tarefas como reconhecimento de escrita, voz e objetos, além de tradução de textos e séries temporais (AGGARWAL, 2018).

2.2.2.5 Redes Long-Short Term Memory

A rede *Long-Short Term Memory* (LSTM) é um tipo de rede recorrente, introduzida por Hochreiter & Schmidhuber (1997) para solucionar o problema de desaparecimento do gradiente encontrado nas redes recorrentes tradicionais (OSINGA, 2018). A LSTM é uma célula de memória capaz de guardar informações de longo prazo e de selecionar quais informações devem ser esquecidas, conforme Greff et al. (2015).

A célula de memória de uma LSTM funciona através de três portas, ou seja, a porta de entrada (i_t) responsável por adicionar informações na memória; a porta do esquecimento (f_t) para liberar informações da memória e a porta de saída (o_t) para ler informações da memória. Cada célula possui três entradas e duas saídas conforme mostrado na Figura 11 (HOCHREITER; SCHMIDHUBER, 1997).

Figura 11 – Estrutura de uma célula LSTM.



Fonte: Li et al., (2017).

O funcionamento da LSTM pode ser descrito pelas Equações 2.16 a 2.21. Em que a entrada de dados na célula é representada pelo vetor x_t, h_{t-1} representa a saída anterior da célula, W_f, W_i, W_o, W_c e b_f, b_i, b_o e b_C representam respectivamente os pesos e bias das portas de esquecimento, entrada, saída e do estado da célula atual (C_t). Todas as portas utilizam uma função de ativação sigmoide (σ). $\tilde{C}_t$ corresponde ao vetor candidato no instante *t* que será adicionado ao estado da célula (C_t) (GREFF et al., 2015; SONG; HUANG; SONG, 2019).

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{2.16}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.17}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.18}$$

$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{2.19}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{2.20}$$

$$h_t = o_t * \tanh(C_t) \tag{2.21}$$

2.2.2.6 Formas de aprendizado

As redes neurais artificiais possuem uma característica primordial em aprender por meio de exemplos. Este aprendizado é um processo que realiza ajustes dos pesos (BRAGA; CARVALHO; LUDERMIR, 2007; HAYKIN, 2001). Os principais modelos de aprendizado são denominados supervisionado e não-supervisionado.

No aprendizado supervisionado, existe um supervisor no processo de aprendizagem responsável por determinar a entrada e observar a saída. De acordo com Braga, Carvalho e Ludermir (2007), para cada entrada fornecida à rede, a sua saída é comparada com a saída desejada pelo supervisor. Segundo Lecun, Bengio e Hinton (2015), este modelo é a forma mais usual de aprendizado de máquina.

No modelo não supervisionado, não existe uma saída desejada para ser comparada com a saída calculada pela rede. Neste caso, somente as entradas são fornecidas à rede, não existindo um supervisor para fornecer resposta para cada processo do aprendizado conforme Haykin (2001). Este modelo é usualmente aplicado em problemas que envolvem descoberta de agrupamentos ou classes (BRAGA; CARVALHO; LUDERMIR, 2007).

2.2.2.7 Algoritmos de treinamento

Os algoritmos de treinamento ou otimizadores são responsáveis por realizar os ajustes nos pesos da rede neural artificial. O algoritmo de aprendizagem supervisionada mais popular para o treinamento de redes neurais multicamadas foi proposto por Rumelhart, Hinton e Willians (1986), conhecido como *backpropagation*.

O *backpropagation* baseia-se na regra delta proposta por Widrow e Hoff (1960), composto por duas fases, ou seja, a fase para frente (*forward*) e a fase para trás (*backward*) (Faceli et al., 2011). Na fase *forward*, as entradas são apresentadas a rede, e os sinais são propagados para frente com a finalidade de calcular a saída. Na fase *backward*, é comparada a saída da rede com a saída desejada, e os pesos são atualizados (AGGARWAL, 2018; BRAGA; CARVALHO; LUDERMIR, 2007).

As Equações 2.22 a 2.25 definem como serão calculados os pesos de uma rede neural multicamadas (FACELI et al., 2011).

$$e_l = \frac{1}{2}\sum_{k=1}^{n}(yd_l(k) - y_l(k))^2 \quad (2.22)$$

$$w_{jl}(n+1) = w_{jl}(n) + \alpha x^j \delta_l \quad (2.23)$$

$$\delta_{l\ camada\ saida} = f_a' e_l \quad (2.24)$$

$$\delta_{l\ camada\ oculta} = f_a' \sum w_{lk}\delta_k \quad (2.25)$$

Para realizar a atualização dos pesos, é necessário calcular o erro da camada de saída conforme Equação 2.22, em que yd_j representa a saída desejada do neurônio (*l*), y_l o valor de saída e 'n' o número de neurônios. Após o cálculo do erro (e_l), é feito a atualização dos pesos na fase *backward* (EQUAÇÃO 2.23). Ainda na Equação 2.23, temos que os pesos de um neurônio (*l*) de um elemento de entrada ou saída (*j*) são representados por w_{jl}, a taxa de aprendizagem por α e o gradiente por δ_l.

O cálculo do gradiente (δ_l) é feito de forma diferente para cada tipo de camada (saída e oculta). Os gradientes das camadas de saída($\delta_{l\ camada\ saida}$) e oculta ($\delta_{l\ camada\ oculta}$) são calculados conforme as Equações 2.24 e 2.25. Nestas equações, $f_a^{'}$ representa a derivada parcial da função de ativação do neurônio e e_l o erro quadrático obtido na camada de saída (FACELI et al., 2011).

O algoritmo *backpropagation* possui algumas dificuldades de treinamento, entre elas está a difícil convergência em regiões de mínimos locais, para evitar essas regiões faz-se a utilização do termo *momentum* (β) conforme Equação 2.26 (HAYKIN, 2001; RUMELHART; HINTON; WILLIANS, 1986).

$$w_{jl}(n+1) = w_{jl}(n) + \alpha x^j \delta_l + \beta(w_{jl}(n) - w_{jl}(n-1)) \quad (2.26)$$

2.2.3 Floresta Aleatória

O algoritmo de Florestas Aleatórias (*Random Forest*) pode ser utilizado em tarefas de classificação e regressão, pois, possuem alto desempenho com apenas alguns parâmetros para ajustes (GENUER et al., 2017). Esse algoritmo pode ser aplicado em banco de dados amplos

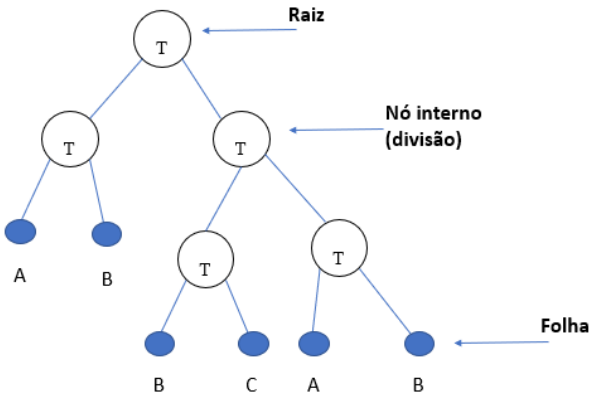
de diversas áreas, dentre elas: sensoriamento remoto, diagnóstico médico por imagem, predição rápida de movimentos, etc. (BELGIU; DRAGUT, 2016; CRIMINISI et al., 2013; GENUER et al., 2017; SHOTTON et al., 2013).

O termo Floresta Aleatória foi descrito em 2001 por Leo Breiman. É definido como um algoritmo de aprendizagem de máquina que combina várias árvores de decisão, geradas de forma aleatória a partir da amostra original (BREIMAN, 2001). Além disso, combina os resultados para alcançar um melhor poder preditivo dessas árvores separadamente (BIAU; SCORNET, 2016). De acordo com Dietterich (2000), esta abordagem de combinação de vários classificadores ou regressores para predizer um novo exemplo é chamada de *Ensemble*.

As árvores de decisão podem ser denominadas como modelos estatísticos que através de um treinamento supervisionado são capazes de prever dados e classificá-los. Para arquitetar uma árvore de decisão, os treinamentos são feitos através de entradas e saídas. O princípio das árvores de decisão é ‘dividir para conquistar’, ou seja, um problema complexo é separado em problemas mais simples (CREPALDI, 2012).

Desta forma, as árvores de decisão são formadas por nós, ramos (nós internos) e folhas que representam respectivamente todos os atributos, possíveis valores para os atributos e as distintas classes. Portanto, o conjunto de dados é classificado subdividindo-o sequencialmente de acordo com as estruturas de decisão definidas pela árvore (FIGURA 12) (FRIEDL; BRODLEY, 1997).

Figura 12 – Estrutura da árvore de decisão.



Fonte: Adaptado de Friedl e Brodley (1997).

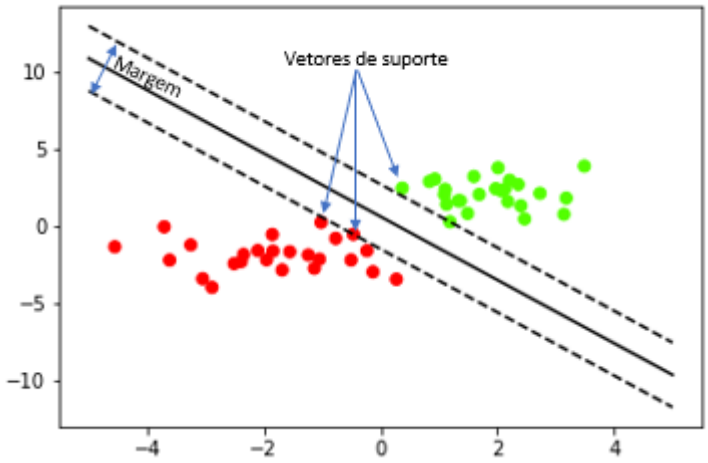
Desta maneira, os pontos positivos que influenciam na escolha do algoritmo de Florestas Aleatórias são a capacidade de suportar dados de elevada dimensionalidade de maneira satisfatória, além de ser possível trabalhar com dados ausentes e ainda ter precisão.

2.2.4 Máquina de Vetores de Suporte

A Máquina de Vetores de Suporte (SVM) é um modelo de aprendizado supervisionado que pode ser utilizado para resolver problemas de classificação e regressão. Sua teoria é baseada na teoria de aprendizado estatístico conforme Cortes e Vapnik (1995).

Em problemas linearmente separáveis o algoritmo busca encontrar o melhor hiperplano que separa as classes. O hiperplano é definido pelos vetores de suporte localizados na margem de decisão. Desta forma, o melhor hiperplano é determinado pela direção que maximiza a margem de decisão (FIGURA 13) (LUGER, 2013; RUSSEL; NORVIG, 2012; SALHI; TARI; KECHADI, 2021).

Figura 13 – Hiperplano separador de duas classes.



Fonte: Do Autor (2019).

Para problemas não linearmente separáveis, o SVM transforma o conjunto de dados de entrada do espaço original em um novo espaço de maior dimensão, conhecido como espaço de características, ou também de função kernel. Desta forma, os dados se tornam linearmente separáveis (FACELI et al., 2011; RUSSEL; NORVIG, 2013; SUYKENS; VANDEWALLE, 1999).

2.2.5 K vizinhos mais próximos

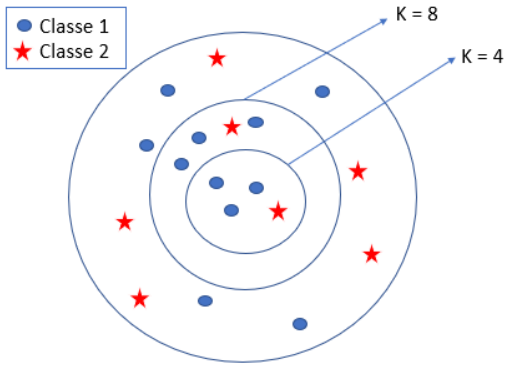
O K vizinhos mais próximos (*K-Nearest Neighbor* - KNN) é um algoritmo de aprendizado supervisionado proposto por Aha, Kibler e Albert (1991). No KNN, os objetos são classificados com base no conjunto de treinamento. Procura-se K elementos a partir do

conjunto de treinamento que estejam mais próximos do elemento desconhecido a ser classificado. Desta maneira, o novo elemento é classificado como a classe majoritária dos K vizinhos mais próximo (FIGURA 14) (WETTSCHERECK; DIETTERICH, 1994).

Em problemas de regressão, pode-se utilizar duas abordagens: minimizar o erro quadrático, através da média dos valores encontrados para os k vizinhos, ou minimizar a função custo através do desvio absoluto, por meio da mediana (FACELI et al., 2011).

O treinamento do algoritmo KNN é baseado no cálculo da distância entre os objetos. Quanto menor a distância, mais similaridade terão e desta forma, maior a chance de classificação do elemento desconhecido (GUO et al., 2003).

Figura 14 – Exemplo de classificação KNN.



Fonte: Adaptado de Pacheco (2017).

O único parâmetro a ser definido pelo algoritmo é o número de vizinhos mais próximos (K). Usualmente este valor é pequeno e ímpar, para evitar empates. Entretanto, não existe uma regra para se definir o valor de K e assim o valor pode ser definido por tentativa e erro (CASTRO; FERRARI, 2016; FACELI et al., 2011).

2.3 Séries Temporais

Uma série temporal, ou série histórica, é um conjunto de dados gerados sequencialmente no tempo (JIANGUANG; JIRUTITIJAROEN, 2010; MORETTIN; TOLOI, 2006). O objetivo da previsão das séries temporais é prever algum valor futuro baseado em amostras de dados passados e atuais, segundo Sapankevych e Sankar (2009). O objetivo de realizar previsões é importante no aprendizado de máquina e possui várias aplicações científicas e tecnológicas. Os objetivos das séries temporais podem estar também relacionados

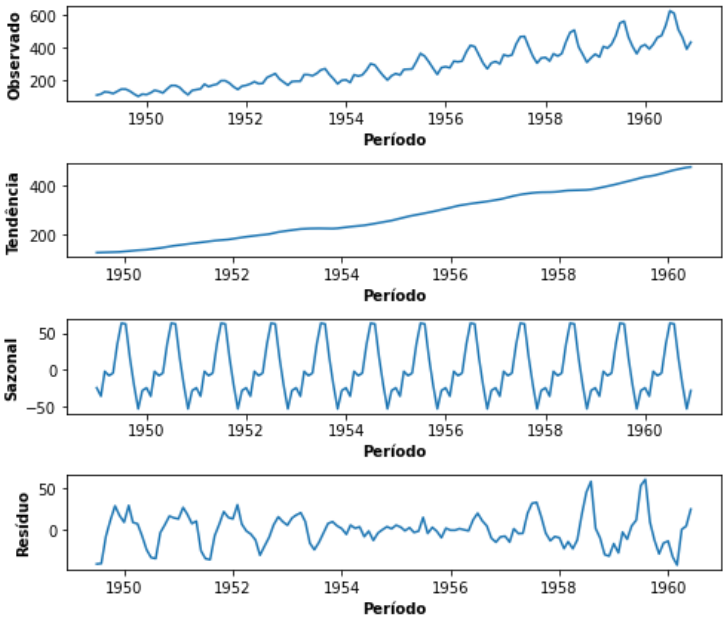
em descrever o comportamento da série e procurar por periodicidade nos dados, de acordo com Morettin e Toloi (2006).

O conjunto de dados de uma série temporal geralmente possui grande volume e é coletado em intervalos de forma anual, mensal, semestral e diário, dentre outras (CORRAR; THEÓPHILO, 2004). Sendo assim, o conjunto de dados pode ser desenvolvido ao longo do tempo em torno de uma média constante, que caracteriza a série temporal como estacionária, caso contrário, ela é definida como não estacionária (MORETTIN; TOLOI, 2006).

Nesse sentido, a caracterização das séries temporais pode ser de ordem univariada quando baseada em observações únicas, ou seja, quando envolve uma única variável, e de ordem multivariada, quando se baseia em mais de uma variável (CHATFIELD, 1988).

O desenvolvimento da variável ao longo do tempo pode ser influenciado por fatores tais como mudanças de padrões e fenômenos imprevisíveis (CORRAR; THEÓPHILO, 2004). A análise de uma série temporal, de acordo com Brockwell e Davis (2002), inicia-se com a plotagem do gráfico referente aos dados, e se houver descontinuidades ou observações distantes torna-se necessária uma verificação. Desta forma, identificando descontinuidades pode ser necessário decompor a série temporal em componentes denominados tendência, sazonalidade e componente aleatório (resíduo) (FIGURA 15) (WEIGEND; GERSHENFELD, 1993).

Figura 15 – Componentes da série temporal.



Fonte: Do Autor (2019).

A tendência é o componente que indica o comportamento da série temporal e exibe as mudanças no nível médio da série em longo prazo. Esse componente pode ser linear ou não-linear e apresenta crescimento ou decrescimento nos padrões e quando há ausência de ambos, a série é denominada estacionária (BOUZADA, 2012; YAFFE; MCGEE, 2002). As movimentações e desvios que podem ser manifestados em torno da tendência são denominados de ciclo ou ciclicidade (BOUZADA, 2012).

A sazonalidade representa as variações de queda ou subida nos dados e possui um comportamento que pode ser repetido em *k* períodos ao longo da série temporal (EHLERS, 2009). Um exemplo da sazonalidade pode ser descrito nas vendas realizadas por lojas e supermercados em períodos de férias, em que o consumo de alguns produtos tende a aumentar e, consequentemente, forma picos em determinados pontos da série temporal de acordo Gurajati e Porter (2011).

Ao extrair os componentes sazonais, cíclicos e de tendência da série temporal, o restante é denominado componente aleatório, conhecido também como termo errático ou resíduo, conforme Yaffe e McGee (2002). Essa aleatoriedade é um movimento irregular, proveniente de um fenômeno imprevisível (BOUZADA, 2012).

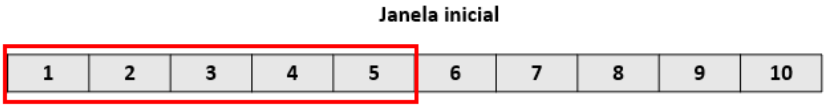
2.3.1 Processamento Temporal

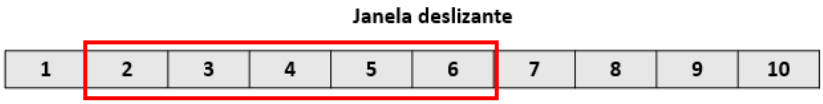
Uma das principais aplicações de algoritmos de aprendizado de máquina é o processamento temporal, em que o tempo é um parâmetro importante. Para incorporar uma estrutura temporal para problemas de aprendizado supervisionado, pode-se utilizar o método da janela deslizante (BRAGA; CARVALHO; LUDERMIR, 2007; DIETTERICH, 2002).

O método da janela deslizante consiste em converter as sequências de entradas e saída em um conjunto de janelas (h_w). Portanto, é necessário definir uma largura *d* da janela. Assim, para h_w prever a saída atual (y_i) no tempo (*t)* é utilizada a janela ($x_{i,t-d,}$ $x_{i,t-d+1,...,}x_{i,t+d-1,}x_{i,t+d}$). Desta forma, através dessa conversão, é possível aplicar qualquer algoritmo de aprendizado supervisionado (DIETTERICH, 2002; JOSHI; DIETTERICH, 2003).

Após definir a largura da janela (*d)* e horizonte de previsão, converter os dados em sequências de pares (entrada e saída) em conjunto de janelas, o processo de previsão será realizado até que todos os conjuntos sejam utilizados (FIGURA 16) (HOTA; HANDA; SHRIVAS, 2017).

Figura 16 – Exemplo de janela deslizante com largura = 5 para previsão de um passo à frente.





Dados reestruturados

Xi(t-5)	*Xi(t-4)*	*Xi(t-3)*	*Xi(t-2)*	*Xi(t-1)*	*y(t)*
1	2	3	4	5	6
2	3	4	5	6	7
3	4	5	6	7	8
4	5	6	7	8	9
5	6	7	8	9	10

Fonte: Adaptado de Hota, Handa, Shrivas (2017).

2.3.2 Índice Nacional de Preços ao Consumidor Amplo (IPCA)

O Índice Nacional de Preços ao Consumidor Amplo (IPCA) é um índice que pode ser utilizado para inflacionar séries temporais de preços de produtos e serviços (CARRARA; CORREA, 2012).

O IPCA é calculado pelo Instituto Brasileiro de Geografia e Estatística (IBGE), com o objetivo de medir a inflação de determinados grupos de produtos e serviços no varejo, abrangendo como cenário de pesquisa famílias com renda entre 1 e 40 salários-mínimos (IBGE, 2019; NOGAMI; PASSOS, 2016).

O IPCA pode ser agregado em níveis de grupos, compostos por categorias como: alimentação e bebidas, saúde, transportes, habitação, educação, vestuário, comunicação, artigos de residência e despesas pessoais (MARTINEZ; CERQUEIRA, 2013).

O IPCA é obtido através de coletas de informações, conhecidas como Pesquisas de Orçamento Familiar (POF), realizadas nas regiões metropolitanas de Belém, Recife, Fortaleza, Salvador, Rio de Janeiro, São Paulo, Belo Horizonte, Porto Alegre, Curitiba, Brasília e município de Goiânia, com o propósito de investigar os hábitos de consumo mais comuns da população. Desta forma, através das POF's são determinados os pesos de cada item nos gastos do consumidor, e os pesos de cada categoria são resultados da soma dos pesos dos itens que a compõe (MARIANO, 2012; MARTINEZ; CERQUEIRA, 2013; MELO, 1986).

2.3.3 Métricas de avaliação

As métricas de avaliação são elementos importantes no aprendizado de máquina, pois, através delas é possível avaliar e estimar o quanto o modelo está errado. As métricas, erro médio quadrático (MSE), erro médio absoluto (MAE), raiz quadrada do erro médio quadrático (RMSE) e coeficiente de determinação (R^2) são as mais comumente utilizadas em modelos de previsão (HUYNH et al., 2020). A raiz do erro logarítmico quadrático médio (RMSLE) também é um bom indicador para previsão, pois, ele serve para avaliar o desempenho das previsões, de acordo com Salaken et al. (2015).

O erro médio quadrático (MSE) calcula a média dos erros ao quadrado. Desta forma, julga os diferentes graus entre os valores previstos e reais. Quanto menor o MSE, melhor o previsor (GUO et al., 2015). O MSE pode ser calculado conforme Equação 2.27, onde y_i representa o valor real, $\tilde{y}_i$ o valor previsto e n o número de elementos.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2 \tag{2.27}$$

O RMSE é uma métrica que possui apenas valores positivos, pois os erros são elevados ao quadrado antes de realizar a média. Desta forma, maiores erros terão uma maior penalização (GUO et al., 2015; SALAKEN et al., 2015). O RMSE pode ser expresso conforme Equação 2.28. Em que y_i representa o valor observado, $\tilde{y}_i$ o valor previsto e n o número de amostras.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2} \tag{2.28}$$

A raiz do erro logarítmico quadrático médio (RMSLE) é uma métrica que ao contrário do RMSE, atribui uma penalização maior aos erros nas quais as previsões deveriam ser mais próximas de zero (SALAKEN et al., 2015). O RMSLE é descrito na Equação 2.29.

$$\text{RMSLE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(y_i + 1) - \log(\tilde{y}_i + 1))^2} \tag{2.29}$$

O erro médio absoluto (MAE) é uma métrica que dá o mesmo peso para todas as diferenças entre o valor real e o valor previsto. Pode ser expresso conforme Equação 2.30.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \tilde{y}_i| \tag{2.30}$$

Além das métricas descritas anteriormente, pode ser aplicado também o coeficiente de determinação (R^2), que se refere a proporção de variação da variável dependente (explicada), pela variável independente (RAWLINGS; PANTULA; DICKEY, 1998). O resultado do R^2 deve ser avaliado com cautela, sendo que valores grandes podem indicar que o modelo se ajustou bem aos dados, por outro lado, valores pequenos pode indicar que o modelo não se adequou aos dados (MONTGOMERY; RUNGER, 2013).

O R^2 é calculado conforme Equação 2.33, em que o somatório dos quadrados do resíduo *SS(Res)* pode ser calculado de acordo com a Equação 2.31, e a soma total dos quadrados $SS(Total)$ calculado conforme a Equação 2.32. Desta forma, temos que y_i representa o valor observado, $\hat{y}_i$ o valor previsto, $\bar{y}_i$ a média e *n* o número de amostras.

$$SS(Res) = \sum_{j=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.31}$$

$$SS(Total) = \sum_{j=1}^{n}(y_i - \bar{y}_i)^2 \tag{2.32}$$

$$R^2 = 1 - \frac{SS(Res)}{SS(Total)} \tag{2.33}$$

2.4 Seleção de características

Algumas bases de dados podem possuir muitas características (atributos, variáveis), mas, uma parte pode ser irrelevante ou redundante para o conjunto de dados (FACELI et al., 2011). O objetivo da seleção de características ou atributos é selecionar um subconjunto a partir de um conjunto de dados, com a finalidade de reduzir características irrelevantes, facilitar a visualização e compreensão dos dados, evitar a maldição da dimensionalidade, obter um melhor desempenho na execução dos algoritmos de aprendizado e consequentemente fornecer bons resultados de previsão (CHANDRASHEKAR; SAHIN, 2014; GUYON; ELISSEEFF, 2003).

Existem várias técnicas que se aplicam a seleção de características e visam encontrar o subconjunto ótimo de características. Na avaliação do desempenho do subconjunto podem-se utilizar três abordagens: a Embutida (*Embedded*), o Filtro e o *Wrapper* (FACELI et al., 2011).

Na abordagem baseada em filtro, a finalidade como o próprio nome sugere é filtrar ou fazer uma heurística na busca de atributos irrelevantes do conjunto, através de algum critério, antes de aplicar o algoritmo de aprendizagem de máquina (JOHH; KOHAVI; PLEGER, 1994). Sendo assim, o algoritmo de aprendizagem receberá como entrada os atributos relevantes fornecidos pela saída do filtro. A vantagem de se utilizar técnicas de filtragem, de acordo com Saeys, Inza e Larrañaga (2007), é que elas são computacionalmente rápidas, simples e independentes dos algoritmos de aprendizagem.

A abordagem baseada em *wrapper* utiliza o algoritmo de aprendizado de máquina para selecionar e avaliar diferentes subconjuntos de características do conjunto de treinamento. Desta forma, o subconjunto que obtiver um melhor desempenho no processo de aprendizagem é definido como o subconjunto de características (STÁNCZYK; JAIN, 2015).

Na abordagem embutida, a seleção de características é parte interna e comum do algoritmo de aprendizado. As árvores de decisão são um exemplo de algoritmo de aprendizado que utiliza esse método para seleção de atributos (FACELI et al., 2011).

2.5 Trabalhos Relacionados

Redes Neurais Artificiais MLP foram comparadas as técnicas *Naïve* não ajustada e regressão linear, para previsão de vendas de produtos do varejo. Almeida e Passari (2006) realizaram em seu trabalho a seleção de produtos através da técnica de análise de cesto de compras, com objetivo de separar um grupo de produtos que faziam parte de um mesmo centro de interesse de vendas, além de identificar os itens que foram vendidos em uma mesma operação. As RNA's obtiveram um desempenho superior quando comparada aos demais modelos na previsão de vendas de produtos individuais.

Luo et al. (2010) utilizaram os modelos de rede neural *backpropagation*, rede neural com algoritmo genético, rede neural RBF e uma integração de todos para prever o preço de mercado atacadista de vegetais. Os modelos foram comparados através do erro médio absoluto (MAE), obtendo como pior desempenho a rede neural *backpropagation*, e o melhor modelo a combinação de todos.

Hadavandi, Shavandi e Ghanbari (2011) criaram um sistema especialista para a previsão de vendas de placa de circuito impresso (PCB), os autores aplicaram um sistema *fuzzy* com algoritmos genéticos. Entretanto, antes desta etapa, eles utilizaram a técnica de agrupamento de dados, através do algoritmo *K-means* para uma categorização das vendas. O sistema *fuzzy* genético foi avaliado utilizando as métricas de erro percentual absoluto médio (MAPE) e o RMSE, e comparada com outros trabalhos que utilizaram *fuzzy* integrados a técnicas de aprendizado de máquina, como redes neurais artificiais. Os resultados obtidos pelo sistema *fuzzy* genético superaram as abordagens anteriores em ambas métricas avaliadas.

Os valores de vendas do varejo no *Walmart* foram previstos por Catal et al. (2019) através dos métodos de regressão de aprendizado de máquina (Regressão linear, Rede Neural, Árvore de decisão e Floresta aleatória). Neste trabalho, buscou-se comparar as técnicas clássicas de análise de séries temporais (ARIMA e média móvel) que foram aplicadas e avaliadas em dois cenários diferentes. Inicialmente os modelos foram aplicados em apenas uma loja e departamento da empresa varejista para avaliação, e posteriormente estendidos para todos os departamentos. Os resultados foram avaliados através das métricas RMSE, MAE e R^2. O modelo de floresta aleatória alcançou um melhor resultado em todas as métricas e em ambos os cenários propostos.

Spiliotis et al. (2020) realizaram previsões de demanda diária de 3300 bens de consumo para uma empresa varejista aplicando as técnicas estatísticas *Croston* e suas variantes. Para tanto, uma comparação em relação ao desempenho da previsão com métodos

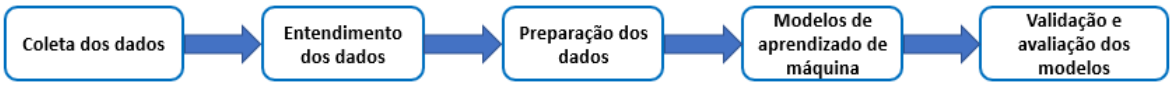
de aprendizado de máquina (SVR, KNN, Floresta aleatória, Redes Neurais e *Gradient Boosted Tree*, dentre outras) foi realizada. O desempenho da previsão de demanda dos produtos foi avaliado através das métricas *Mean Absolute Scaled* Error (AMSE) e *Root Mean Squared Scaled Error* (RMSSE). Os resultados experimentais mostraram que os algoritmos de aprendizado de máquina superaram os métodos clássicos.

Modelos de regressão linear e floresta aleatória foram utilizados por Awan et al. (2021) em um conjunto de dados com 0,55 milhões de observações de vendas da *Black Friday* para previsão de vendas e também para auxiliar empresas do varejo no intuito de criar promoções personalizadas para seus clientes. Os modelos aplicados foram implementados em estrutra Spark (própria para grandes conjuntos de dados) e comparada com uma estrutura que não apropriada para *Big Data*. Neste trabalho, o modelo de floresta aleatória obteve o melhor resultado em ambas as estruturas implementadas.

3 MATERIAL E MÉTODOS

Neste capítulo, são abordados materiais, ferramentas, métodos e etapas aplicadas para realização desta dissertação. O desenvolvimento deste trabalho foi realizado inicialmente com a coleta, estatística descritiva, análise exploratória e transformação dos dados. Posteriormente, aplicação e validação dos modelos de aprendizado de máquina e, por fim, avaliação dos resultados dos modelos (FIGURA 17) (CUSTA & KUMAR, 2016).

Figura 17 – Esquema da metodologia proposta.



Fonte: Adaptado de (CUESTA; KAMAR, 2016).

3.1 Coleta dos dados

Os dados que foram coletados eram provenientes de uma empresa de varejo do segmento de supermercados localizada na cidade de Cambuí/MG. Desta forma, por pertencerem ao setor privado o acesso a essas informações foram mantidas restritas.

A seleção dos dados foi realizada a partir dos 20 produtos mais vendidos (TABELA 1) para a obtenção da série histórica de vendas diárias de segunda-feira a domingo (abril 2015 a dezembro 2019), totalizando em 34.646 registros coletados das vendas. As vendas destes produtos forneceram uma base de dados de séries temporais com seis atributos (TABELA 1), sendo eles:

- Data: Definida por dia, mês e ano em que ocorreu a venda;
- Id_produto: Identificação única do produto utilizado internamente no sistema da rede varejista;
- Descrição: Nome por extenso do produto. É utilizada para identificar e descrever um produto;
- Custo (R$): Representa o valor unitário pago ao fornecedor pela rede varejista para aquisição do produto;
- Preço (R$): Representa o valor unitário de um produto utilizado na venda; e
- Vendas: Valores em quantidades de saída dos produtos.

Tabela 1 – Listagem dos produtos.

Id_produto	Descrição	Data	Custo	Preço	Vendas
67	Azeitona granel	15/10/2019	12,66	14,11	4
1722	Açúcar	07/05/2019	2,22	2,99	3
1835	Água Mineral	12/05/2016	1,00	1,37	1
2542	Arroz	01/04/2015	9,24	11,90	19
5006	Extrato de tomate A	13/10/2019	3,18	4,06	1
6369	Cerveja A	25/09/2019	2,10	2,43	8
6388	Cerveja B	02/01/2019	2,12	2,59	20
6426	Cerveja C	01/03/2019	2,33	2,47	22
11689	Extrato de tomate B	05/10/2018	1,58	2,28	1
11917	Farinha de trigo	04/06/2019	2,34	3,15	6
11976	Feijão	04/10/2019	4,10	5,49	4
15416	Leite desnatado	01/04/2015	1,40	1,79	31
15417	Leite integral	16/04/2019	2,30	2,75	11
18196	Óleo de soja	01/04/2015	2,37	2,99	67
19852	Polvilho	20/11/2019	5,40	6,06	1
20833	Refrigerante A	09/04/2019	4,28	5,81	4
20865	Refrigerante B	10/10/2015	4,84	5,01	11
20869	Refrigerante C	10/11/2019	5,83	6,50	13
22288	Sal	05/03/2018	1,95	2,74	4
25309	Pacote de tempero	12/07/2019	2,64	3,45	1

3.2 Entendimento dos dados

A estatística descritiva e a análise exploratória dos dados foram realizadas através dos gráficos gerados pela linguagem de programação *Python*™ (versão 3.7.4) [1]. As bibliotecas utilizadas para gerar esses gráficos foram: *matplotlib* (3.1.1) [2] e *seaborn* (0.9.1) [3].

Ao analisar a estatística descritiva e a análise exploratória desse conjunto de dados, foi possível extrair as características intrínsecas deles, encontrar variáveis correlacionadas com a variável saída, além de permitir identificar visualmente as distribuições e as tendências, sendo estas últimas estimadas pelo ajuste de regressão linear (BROCKWELL; DAVIS, 2002; PAL; PRAKASH, 2017).

[1] https://www.python.org/
[2] https://matplotlib.org/
[3] https://seaborn.pydata.org/

Verificou-se a sequência temporal de vendas visando identificar e corrigir prováveis falhas na cronologia dos dados, deixando assim as séries temporais igualmente espaçadas no tempo, para dessa forma ser possível realizar a aplicabilidade dos modelos preditivos (BAHADORI; LIU, 2012).

Para tornar as séries temporais igualmente espaçadas, devido a falhas de períodos omitidos de vendas, foram utilizadas as seguintes abordagens (FIGURA 18):

- Em decorrência de ausência de saída do produto, o valor do atributo “vendas” foi considerado como zero;
- Os atributos “custo” e “preço”, foram preenchidos com o último valor válido observado.

Figura 18 – Exemplo de correção de falhas de períodos omitidos de vendas.

Data	id_produto	Descrição	Vendas	Custo (R$)	Preço (R$)
03/01/2018	1722	Açúcar	4	2,58	3,57
04/01/2018	1722	Açúcar	Sem informações de vendas, preço e custo		
27/12/2018	18196	Óleo de soja	8	2,72	3,64
28/12/2018	18196	Óleo de soja	Sem informações de vendas, preço e custo		
29/12/2018	18196	Óleo de soja	3	2,72	3,64

Data	id_produto	Descrição	Vendas	Custo (R$)	Preço (R$)
03/01/2018	1722	Açúcar	4	2,58	3,57
04/01/2018	1722	Açúcar	0	2,58	3,57
27/12/2018	18196	Óleo de soja	8	2,72	3,64
28/12/2018	18196	Óleo de soja	0	2,72	3,64
29/12/2018	18196	Óleo de soja	3	2,72	3,64

Fonte: Do autor (2021).

Para cada modelo específico de aprendizado de máquina, as saídas dos dados eram diferentes. Desta maneira, na previsão de preços o atributo de saída do modelo foi denominado “preço” e para demanda, o atributo saída foi chamado de “vendas”.

3.2.1 Inflação do preço e custo

Os atributos que possuíam valores em unidades monetárias (“preço” e “custo”) não sofreram ajustes da inflação em relação ao último período do conjunto de dados (Dezembro/2019). Desta forma, foi necessário inflacionar esses valores de unidades monetárias de acordo com o Índice Nacional de Preços ao Consumidor Amplo (IPCA),

visando reduzir a variabilidade e o componente tendência das séries temporais (IPEA, 2019; NAU, 2020) (TABELAS 2 e 3).

Tabela 2 – Amostra de registros com correção de valores (R$) pela inflação.

Data	Id_produto	Descrição	Custo sem inflação	Custo c/ inflação	Preço sem inflação	Preço c/ inflação
01/04/2015	6426	Cerveja	2,08	2,62	2,89	3,64
16/01/2016	11976	Feijão	3,78	4,48	4,20	4,97
30/10/2017	1722	Açúcar	1,80	2,13	2,03	2,22
09/12/2018	19852	Polvilho	4,56	4,76	5,50	5,75
12/04/2019	20833	Refrigerante A	4,19	4,31	5,69	5,85

Tabela 3 – Amostra de 5 registros do IPCA.

Data	IPCA – geral -índice
2007.12	2.731,6200
2009.11	3.006,0000
2009.12	3.017,5900
2019.03	5.177,4700
2019.04	5.206,9800

Para inflacionar os valores de unidades monetárias, foi aplicado o cálculo de correção de valores que necessita do índice de correção calculado a partir da Equação 2.34. E o valor inflacionado foi calculado de acordo com a Equação 2.35:

$$Índice\ de\ correção = \frac{Índice\ Data\ base}{Índice\ Data\ venda} \tag{2.34}$$

$$Valor\ Corrigido = (x\ .\ Índice\ de\ correção) \tag{2.35}$$

Na Equação 2.34, *x* representa o valor a ser atualizado, *índice data venda* equivale a data em que o produto foi vendido e o *índice data base* é a data base relativa ao valor que será inflacionado, para este último foi considerado o mês de dezembro de 2019.

3.2.2 Correlação entre os atributos

A correlação de *Pearson* com um nível de significância de 5% foi aplicada em ambas as séries temporais (demanda e preço). Através da biblioteca *pandas* (0.25.1) [4] foi possível realizar o cálculo da correlação entre os pares de atributos e a biblioteca *seaborn* (0.9.1) para plotar o gráfico. Por fim, a escala descrita por Mukaka (2012) foi empregada para analisar os coeficientes de correlação calculados entre os atributos, eles podem ser avaliados de maneira positiva ou negativa da seguinte forma:

- 0,9 – Indica uma correlação muito forte;
- 0,7 a 0,9 – Indica uma correlação forte;
- 0,5 a 0,7 – Indica uma correlação moderada;
- 0,3 a 0,5 – Indica uma correlação fraca; e
- 0 a 0,3 – Indica uma correlação muito fraca.

3.3 Preparação dos dados

Nesta etapa, foram descritas as abordagens utilizadas para criação de novos atributos (engenharia de atributos) para os modelos de aprendizado de máquina aplicados, transformações dos dados e seleção dos atributos.

3.3.1 Engenharia de atributos

A partir das datas obtidas do conjunto de dados referentes às vendas, foram separados os atributos: dia (dia específico, nº), mês (1-12), ano (2015-2019), dias da semana (0-6), semanas do ano (1-53), trimestres (1-4) e final de semana (0 ou 1), para assim construir atributos sazonais (TABELA 4).

[4] https://pandas.pydata.org/

Tabela 4 – Amostra de 5 atributos sazonais criados a partir do atributo “data” para as séries temporais de preços e demandas.

Data	Final de semana	Ano	Mês	Dia	Trimestre	Semana do ano	Dia da semana
23/09/2017	1	2017	09	23	3	38	5
11/10/2017	0	2017	10	11	4	41	2
14/02/2018	0	2018	02	14	1	7	2
21/03/2018	0	2018	03	21	1	12	2
19/10/2019	1	2019	10	19	4	42	5

Através dos atributos de saída (“preço” e “vendas”) foi possível obter valores defasados em *n* períodos (*lags*). As defasagens consistem nos valores de “preço” e “vendas” em tempos passados. O período adotado para defasagem desses atributos correspondeu aos últimos 30 dias. Para assim, serem obtidos os novos atributos “preco_lag_x” e “vendas_lag_x”, em que os valores mínimo e máximo de *x* corresponderam ao intervalo entre 1 e 30 dias (TABELAS 5 e 6). Para o modelo obter informações sobre as tendências e as sazonalidades das vendas dos produtos, foram utilizados os atributos defasados (*lags*) (DOGANIS et. al., 2006; TSOUMAKAS, 2019).

Ao realizar as defasagens dos atributos, uma perda das primeiras observações do conjunto de dados foi observada, isso pode ser justificado devido aos valores deslocados serem desconhecidos (TABELA 7) (PETNEHÁZI, 2019).

Os novos atributos foram criados através da biblioteca *pandas* (0.25.1) e para a seleção dos atributos mais relevantes e treinamento dos modelos de aprendizado de máquina o atributo “data” foi removido.

Tabela 5 – Amostra de 5 registros com defasagens do atributo “vendas” para a série temporal de demanda.

data	Vendas	vendas_lag_1	vendas_lag_2	...	vendas_lag_29	vendas_lag_30
01/05/2015	4	14	2	...	10	7
02/05/2015	15	4	14	...	4	10
03/05/2015	3	15	4	...	12	4
04/05/2015	3	3	15	...	2	12
05/05/2015	12	3	3	...	1	2

Tabela 6 – Amostra de 5 registros com defasagens do atributo “preço” para a série temporal de preços.

data	Preço	preço_lag_1	preço_lag_2	...	preço_lag_29	preço_lag_30
10/08/2015	2,85	2,87	2,87	...	2,87	2,87
11/08/2015	2,85	2,85	2,87	...	2,87	2,87
12/08/2015	2,90	2,85	2,85	...	2,87	2,87
13/08/2015	2,90	2,90	2,85	...	2,87	2,87
14/08/2015	2,90	2,90	2,90	...	2,87	2,87

Tabela 7 – Exemplo de registros que foram perdidos devido a defasagem do atributo “vendas”.

data	vendas	vendas_lag_1	vendas_lag_2	...	vendas_lag_29	vendas_lag_30
01/04/2015	7	Valor desconhecido	Valor desconhecido	...	Valor desconhecido	Valor desconhecido
02/04/2015	10	7	Valor desconhecido	...	Valor desconhecido	Valor desconhecido
03/04/2015	4	10	7	...	Valor desconhecido	Valor desconhecido
04/04/2015	12	4	10	...	Valor desconhecido	Valor desconhecido
05/04/2015	2	12	4	...	Valor desconhecido	Valor desconhecido

3.4 Seleção de atributos

Para a seleção de atributos, utilizou-se uma abordagem embutida, através do algoritmo de Floresta aleatória aplicado pela classe *RandomForestRegressor* da biblioteca *scikit-learn* (0.23.0) [5]. Para isso utilizou-se como parâmetro um total de 1.000 árvores, e os demais parâmetros da mesma classe utilizados para identificar a importância dos atributos foram definidos conforme o valor padrão disponibilizado pela própria biblioteca. Além disso, a divisão dos nós das árvores foi possível pela redução da variância, calculada pela Equação 2.36:

$$Variância = \frac{\sum(X - \mu)^2}{N} \tag{2.36}$$

Em que, temos que: X representa o valor observado, μ a média e N a quantidade de elementos observados.

A partir das novas árvores geradas pelo algoritmo, os atributos relevantes selecionados foram os que possuíam menor variância. Ao final deste processo, quanto mais o atributo foi

[5] https://scikit-learn.org/

utilizado na criação das árvores maior era sua importância. E por fim, todos os valores de relevância calculados para os atributos foram multiplicados por 100, a partir disso, os atributos selecionados foram aqueles que a soma cumulativa dos valores de importância fosse até 95% de importância (ISHWARAN; LU, 2018; WOLFE; MICHAUD, 2009).

3.4.1 Transformação dos dados

O conjunto de dados continha atributos com valores de diferentes grandezas, portanto, foram aplicadas distintas técnicas de transformação de dados. Para que os modelos de aprendizagem de máquina K vizinhos mais próximos e Redes Neurais Artificiais apresentassem melhor desempenho, os dados de entrada e saída precisavam ser normalizados para o processo de aprendizagem (CAO; STOJKOVIC; OBRADOVIC, 2016; HAN; KAMBER, 2006; HAYKIN, 2001).

A aplicabilidade de cada uma das técnicas de transformação levou em consideração o tipo de atributo, sendo que para os atributos sazonais devido a sua natureza cíclica (dia, mês, dia da semana, semana do ano e trimestre) os dados foram transformados em uma escala contínua em pares (seno e cosseno) a partir dos cálculos das Equações 2.37 e 2.38. Desta forma, os modelos apresentaram uma simetria para os atributos de natureza cíclica (Tabela 8) (FIGURA 19) (BUSSETI; OSBAND; WONG, 2012; PETNEHÁZI, 2019).

$$\ddot{x} = \operatorname{sen}\left(\frac{2 \,.\, \pi \,.\, x}{\max\,(x)}\right) \tag{2.37}$$

$$\ddot{x} = \cos\left(\frac{2 \,.\, \pi \,.\, x}{\max\,(x)}\right) \tag{2.38}$$

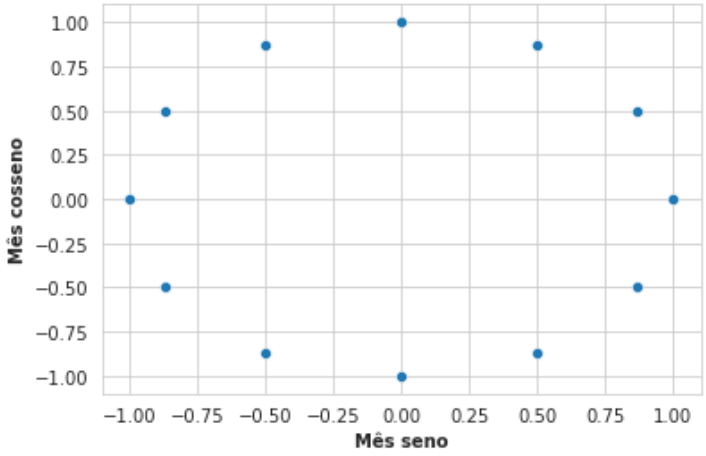
Tabela 8 – Exemplo de transformação cíclica para o atributo “mês” (Continua).

Mês	Mês seno	Mês cosseno
1	0,50	0,87
2	0,87	0,50
3	1,00	0,00
4	0,87	-0,50
5	0,50	-0,87
6	0,00	-1,00
7	-0,50	-0,87
8	-0,87	-0,50
9	-1,00	-0,00

Tabela 8 – Exemplo de transformação cíclica para o atributo “mês” (Conclusão).

Mês	Mês seno	Mês cosseno
10	-0,87	0,50
11	-0,50	0,87
12	-0,00	1,00

Figura 19 – Exemplo de uma transformação cíclica para representação do atributo “mês”.



Fonte: Do autor (2021).

Nos atributos “preço” e “custo” foi aplicada a transformação logarítmica da Equação 2.39 e para o atributo “vendas” a Equação 2.40. Nesta última equação foi acrescentado o valor +1 devido ao atributo “vendas” possuir valores iguais a 0.

As transformações logarítmicas aplicadas acima reduziram a variabilidade dos três atributos (“preço”, “custo” e “vendas”) e os aproximaram de uma distribuição normal. A partir disso foi possível que os modelos de aprendizagem de máquina tivessem um melhor poder preditivo (FENG et. al., 2014; KUHN; JOHNSON, 2019).

E por fim, os atributos “preço”, “custo”, “vendas”, e os atributos com defasagens para “‘vendas” (“vendas_lag_1”, “vendas_lag_2”, ... e “vendas_lag_30”) e “preço” (“preço_lag_1”, “preço_lag_2”, ... e “preço_lag_30”) foram normalizados em um intervalo entre 0 e 1, conforme Equação 2.41.

$$x = \log_{10} x \tag{2.39}$$

$$x = \log_{10}(x + 1) \tag{2.40}$$

$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)} \tag{2.41}$$

3.5 Descrição das ferramentas utilizadas neste trabalho

Na execução deste trabalho, um computador com 6GB de memória RAM, processador Intel Core i5 da 5ª geração de 2.2 GHz e 1 TB de HD foi utilizado para a implementação e execução das técnicas de aprendizado de máquina.

A implementação das técnicas de aprendizado de máquina (Regressão linear, SVM, Redes Neurais Artificiais, Florestas Aleatórias e KNN) aplicadas neste trabalho foram desenvolvidas na linguagem de programação Python™, na versão 3. Além disso, as bibliotecas *scikit-learn* (0.23.0), *Keras* (2.4.3) [6], *Tensorflow* (2.4.1) [7], *Scipy* (1.3.1) [8], *Numpy* (1.16.5) [9], *Orange3* (3.25.1) [10] e *pandas* (0.25.1) foram utilizadas para esta linguagem de programação.

Sobre a linguagem de programação utilizada:

Python™: é uma linguagem de programação interpretada de alto nível, possui um ambiente de desenvolvimento aberto e foi desenvolvido em 1991 por Guido van Rossum (ROSSUM; DRAKE, 2011).

Sobre as bibliotecas utilizadas e suas funções:

- ***scikit-learn***: possui diversos algoritmos, dentre eles, SVM, Florestas aleatórias e *K-means*, dentre outros (PEDREGOSA et al., 2011);
- ***Tensorflow:*** criação e treinamento redes neurais, desenvolvida pela Google®;
- ***Keras:*** (alto nível), executa em *frontend* com o *TensorFlow*, foi desenvolvida para realizar experimentações rápidas, de forma fácil e usual (CHOLLET, 2015);
- ***Scipy:*** fornece algoritmos de otimização, interpolação, equações algébricas e testes estatísticos, dentre outros (VIRTANEN et al., 2020);
- ***Numpy*:** manipulação de dados em vetores e matrizes, além de fornecer também várias funções matemáticas e rotinas de álgebra linear (HARRIS et al., 2020);
- ***Orange3*:** visualização e mineração de dados (DEMSAR et al., 2013); e

[6] https://keras.io/
[7] https://www.tensorflow.org
[8] https://www.scipy.org/
[9] https://numpy.org/
[10] https://pypi.org/project/Orange3/

- ***Pandas*:** trabalha com conjunto de dados estruturados, oferece rotinas integradas para manipular e analisar dados (MCKINNEY, 2011).

3.6 Modelos de aprendizado de máquina

As técnicas de aprendizado de máquina escolhidas para a condução deste trabalho foram a regra do K vizinhos mais próximos ou *K – Nearest Neighbors* (KNN), Floresta Aleatória, Regressão por Vetores de Suporte (SVR), Rede Neural Artificial *Perceptron* multicamadas (MLP), Rede Neural Recorrente *Long Short Term Memory* (LSTM) e Regressão linear.

O KNN foi implementado utilizando a classe *KNeighborsRegressor* da biblioteca *scikit-learn* (0.23.0). Para as previsões de demandas e preços aplicando o KNN, definiu-se o valor de K = 5 para demandas e K = 17 para preços. Os valores de K, que foram selecionados para as previsões de preço e demanda, encontravam-se entre 3 e 99 e foram escolhidos aqueles que obtiveram melhores valores na métrica RMSE (SPILIOTIS et al., 2020). Em ambas as previsões aplicou-se a distância euclidiana.

O modelo de Floresta Aleatória foi implementado utilizando a classe *RandomForestRegressor* da biblioteca *scikit-learn* (0.23.0). Para ambas as séries definidas, o número mínimo de amostras para dividir os nós (*min_samples_split*) foi igual a 2. O número de árvores (*n_estimators*) definidas foi de 1.000, a amostragem por *bootstrap* foi feita com substituição. Foi definido o número mínimo de duas amostras para um ‘nó’ folha, além disso, a profundidade máxima da árvore *(max_depth)* foi determinada conforme a expansão dos “nós” chegasse a um limiar em que o número mínimo de amostras utilizadas para divisão do “nó” fosse menor que a quantidade de amostra do “nó” folha.

O modelo de Regressão por Vetores de Suporte (SVR) foi implementado utilizando a classe SVR da biblioteca *scikit-learn* (0.23.0). Para ambas as séries temporais (preços e demandas), foram aplicadas o *kernel Radial Basis Function* (RBF). Na série das demandas e preços foram atribuídas as penalidades no termo do erro (*C*) de 2.0 e 1.0 respectivamente.

A Rede Neural *perceptron* multicamadas foi implementada utilizando a biblioteca *keras* (2.4.3) e *tensorflow* (2.4.1). Para série temporal das demandas 1.000 épocas de treinamento foram utilizadas, sendo duas camadas ocultas contendo 40 neurônios em cada uma, além de uma regularização de abandono (*Dropout*) com uma taxa de 0,15. As funções de ativações aplicadas foram *ReLU* para as camadas ocultas e linear para a camada de saída.

O algoritmo de otimização para a atualização dos pesos das redes neurais aplicado foi o *Adam* com uma taxa de aprendizagem de 0,001 (KINGMA; BA, 2015). Para série temporal de preços, utilizou-se 500 épocas de treinamento, com duas camadas ocultas contendo 50 neurônios em cada uma e para a regularização de abandono (*Dropout*) definiu-se a taxa de 0,15. As funções de ativações das camadas ocultas foi a *ReLU* e da camada de saída foi a linear. Por fim, o algoritmo de otimização utilizado foi o *Adam* com uma taxa de aprendizagem de 0,001.

A Rede neural *Long Short Term Memory* (LSTM) foi implementada utilizando a biblioteca *keras* (2.4.3) e *tensorflow* (2.4.1). Em ambas as séries foram utilizadas 500 épocas de treinamento, com algoritmo de otimização *Adam* com uma taxa de aprendizado de 0,001, sendo três camadas ocultas, com 50, 40 e 40 neurônios respectivamente. Empregou-se a regularização de abandono com taxa de 0,20 para essas camadas ocultas. As funções de ativação utilizadas nas camadas ocultas foram a *ReLU* e a linear para camada de saída respectivamente.

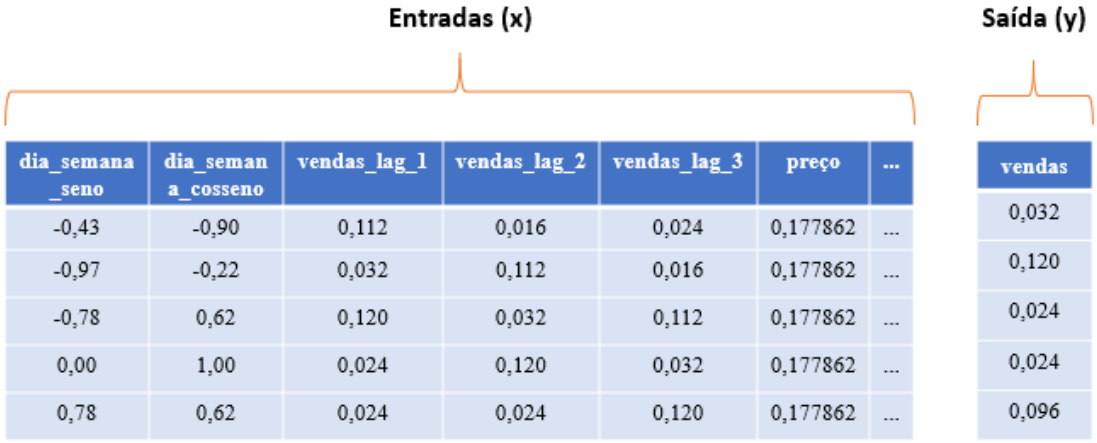
E por fim, o modelo de Regressão Linear foi calculado pelo método dos mínimos quadrados implementado através da biblioteca *scikit-learn* (0.23.0).

3.7 Validação e comparação dos modelos de aprendizado de máquina

Para cada uma das técnicas aplicadas de aprendizagem de máquina, foi avaliada a precisão dos modelos utilizados pelas métricas *Raiz Quadrada do Erro Médio* (RMSE), *Raiz do erro logarítmico quadrático médio* (RMSLE), *Erro Médio Absoluto* (MAE) e *Coeficiente de determinação* (R^2). O tempo computacional despendido para cada modelo de aprendizagem de máquina foi armazenado para posterior comparação. As previsões de preços e demandas foram realizadas diariamente para cada um dos produtos do conjunto de dados.

Para que as previsões dos modelos fossem realizadas, tornou-se necessário que o conjunto de dados fosse convertido em sequências de entradas (x) e saída (y), através da abordagem de janela deslizante (DIETTERICH, 2002). A entrada da janela deslizante foi construída de forma iterativa e composta pelos atributos mais relevantes, sendo eles selecionados pelo algoritmo de Floresta aleatória, para previsão das demandas e preços do dia seguinte (FIGURAS 20 e 21).

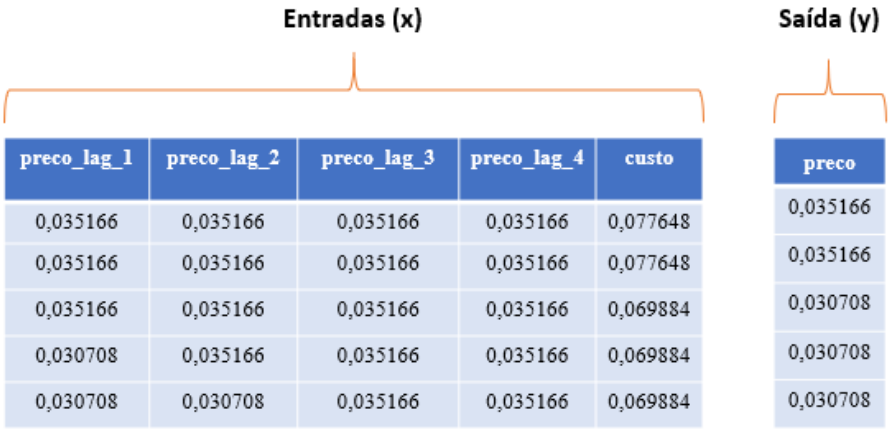
Figura 20 – Exemplo dos dados da série temporal de demandas quando aplicado o supervisionamento de um passo.



dia_semana_seno	dia_semana_cosseno	vendas_lag_1	vendas_lag_2	vendas_lag_3	preço	...	vendas
-0,43	-0,90	0,112	0,016	0,024	0,177862	...	0,032
-0,97	-0,22	0,032	0,112	0,016	0,177862	...	0,120
-0,78	0,62	0,120	0,032	0,112	0,177862	...	0,024
0,00	1,00	0,024	0,120	0,032	0,177862	...	0,024
0,78	0,62	0,024	0,024	0,120	0,177862	...	0,096

Fonte: Do autor (2021).

Figura 21 – Exemplo dos dados da série temporal de preços quando aplicado o supervisionamento de um passo.



preco_lag_1	preco_lag_2	preco_lag_3	preco_lag_4	custo	preco
0,035166	0,035166	0,035166	0,035166	0,077648	0,035166
0,035166	0,035166	0,035166	0,035166	0,077648	0,035166
0,035166	0,035166	0,035166	0,035166	0,069884	0,030708
0,030708	0,035166	0,035166	0,035166	0,069884	0,030708
0,030708	0,030708	0,035166	0,035166	0,069884	0,030708

Fonte: Do autor (2021).

Para que os modelos de aprendizado de máquina fossem validados em diferentes períodos, o conjunto de dados foi dividido em treino e validação. Os tamanhos dos conjuntos foram definidos com um valor de 90 dias para a validação, e para o conjunto de treinamento o tamanho inicial estipulado foi de 546 dias (período 1). Sendo assim, para dar continuidade às validações de previsão dos modelos, após o período 1, o novo conjunto de treinamento era composto pelo valor do conjunto de treinamento anterior somado a 90 dias, como apresentado matematicamente na Equação 2.42. Este processo se repetiu até totalizar 13 períodos subsequentes de validação distintos (FIGURA 22).

Para as técnicas de aprendizagem de máquina que possuem eventos intrinsicamente aleatórios, tais como Redes Neurais Artificiais e Floresta Aleatória, o valor médio do RMSE, RMSLE, MAE e R^2 foi obtido através de 10 execuções de treinamento, para cada um dos 13 períodos de validação.

$$tam.treino = \begin{cases} 546\ dias, & se\ período = 1 \\ treino(período - 1) + 90\ dias, & se\ período > 1 \end{cases} \quad (2.42)$$

Figura 22 – Validação dos modelos.

Fonte: Adaptado de Yang (2018).

Além dos resultados gerados a partir das métricas RMSE, RMSLE, MAE e R^2, um teste de hipótese não paramétrico de *Friedman* (FRIEDMAN,1937) foi utilizado para verificar as diferenças estatísticas nas médias do RMSE dos modelos ao nível de significância de 95%. Para isso, aplicou-se a biblioteca *scipy* (*1.3.1*). Havendo existência de diferenças estatísticas, o teste *Nemenyi* (NEMENYI, 1963) foi empregado para identificar quais os modelos que possuíam médias diferentes, através da biblioteca *Orange3* (3.25.1).

4 RESULTADOS

Neste capítulo são apresentadas a análise exploratória, preparação dos dados e os resultados das técnicas de aprendizado de máquinas, para previsões de preços e demandas. Os códigos-fonte utilizados na preparação dos dados e treinamento dos modelos estão disponíveis no Apêndice A.

4.1 Entendimento dos dados

A rede varejista estudada realiza em média oito operações de venda em um dia, com variações entre 0 e 123 saídas de produto. Desta forma, foi possível inferir que o valor 0 de saída, pode ter ocorrido porque não havia disponibilidade do(s) produto(s) diariamente no estoque ou não foi contabilizado no ponto de venda naquele dia (TABELA 9).

O valor médio dos preços das vendas foi de R$ 5,40, com um mínimo de R$ 1,62 e máximo de R$ 21,81. O valor médio dos custos pagos foi de R$ 4,04, com um mínimo de R$ 0,84 e máximo de R$ 18,87. Em relação ao desvio padrão, observou-se um valor maior (R$ 12,27) que a média das vendas indicando uma alta dispersão entre os valores (TABELA 9).

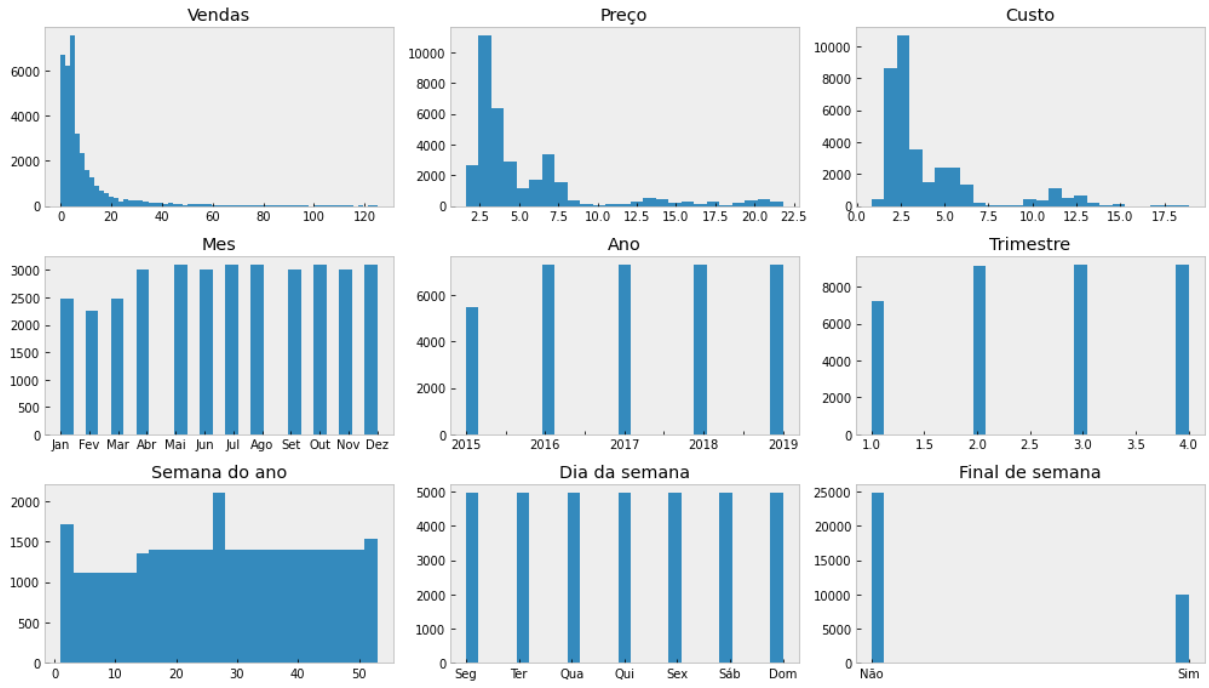
Tabela 9 – Estatística descritiva obtida através da série diária de vendas correspondente aos 20 produtos, relativos ao período de 01/04/2015 a 31/12/2019.

	Preço (R$)	Custo (R$)	Vendas
Média	5,40	4,04	8,59
Desvio padrão	4,21	2,95	12,27
Mínimo	1,62	0,84	0
Máximo	21,81	18,87	123
Mediana	3,63	2,80	4,00
Assimetria	2,24	2,02	3,31

Um decréscimo nas vendas foi observado entre os meses de janeiro, fevereiro e março, e consequentemente no primeiro trimestre durante todo o período definido, isso pode ter ocorrido devido à ausência de registros das vendas anteriores ao mês de abril de 2015. Além disso, no ano de 2015 a ocorrência das vendas, também teve um decréscimo quando comparadas com os demais anos, pelo mesmo motivo da ausência de registros (FIGURA 23). Em relação a assimetria, os atributos descritos na Tabela 9 apresentaram distribuição assimétrica positiva.

Observou-se também uma elevada frequência de produtos sem vendas que equivale a 9% do conjunto de dados, além de uma queda nas vendas aos finais de semana. Em relação ao atributo “Dia da Semana”, observou-se uniformidade na distribuição. Por outro lado, no histograma do atributo “Semana do Ano” os dados são inconclusivos (FIGURA 23).

Figura 23 – Histograma das variáveis.



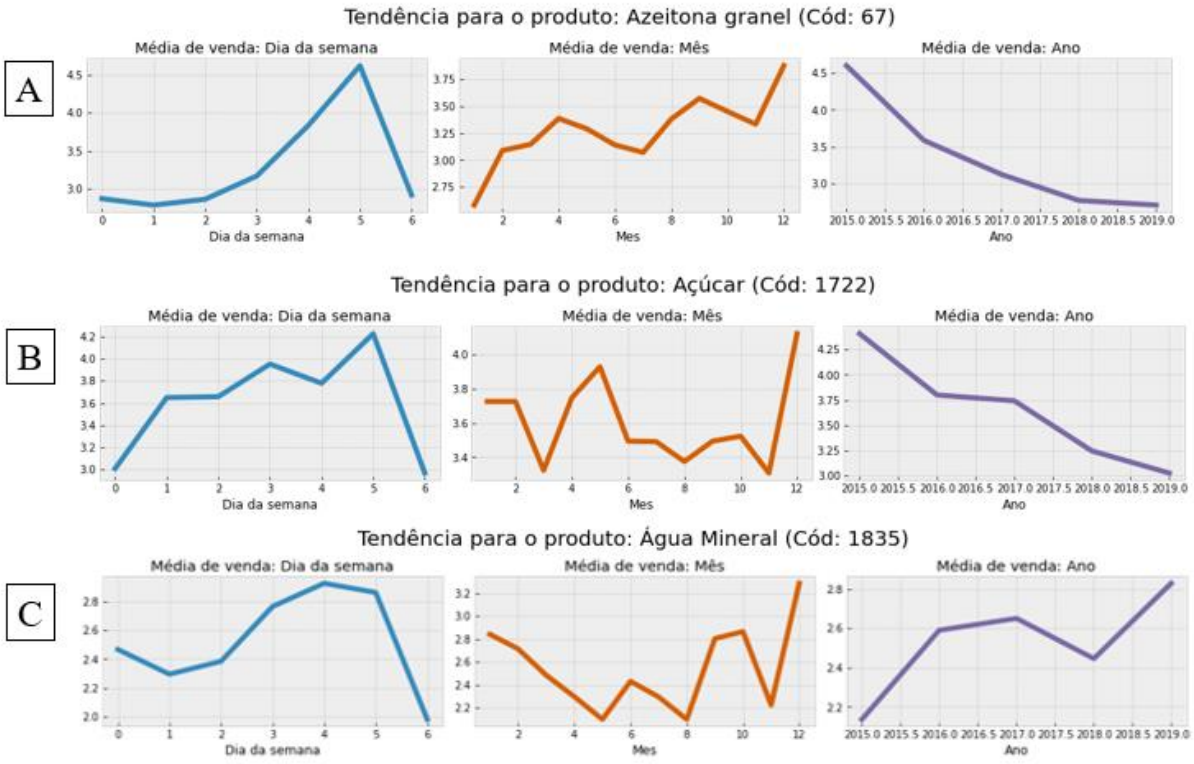
Fonte: Do autor (2021).

4.2 Análise e preparação dos dados para série temporal de demandas

Ao observar o comportamento das vendas de três produtos, que foram agregados pela variável tempo: dias da semana (Seg a Dom), meses (Jan, dez.) e anos (2015-2019), temos que os produtos de código 67 (Azeitona) (FIGURA 24 A) e 1.722 (Açúcar) (FIGURA 24 B) tendem a vender menos com o passar dos anos, o que não ocorre com o produto 1.835 (Água mineral) (FIGURA 24 C). Nos dias de domingo a procura por esses produtos diminui.

A avaliação das vendas mensais dos três produtos descritos anteriormente mostrou que, em cada mês, comportamentos distintos para cada um dos produtos foram detectados, porém, no mês de novembro eles apresentaram quedas nas vendas seguidas de um aumento no mês de dezembro (FIGURA 24), demais produtos apresentaram diferentes tendências (APÊNDICE B).

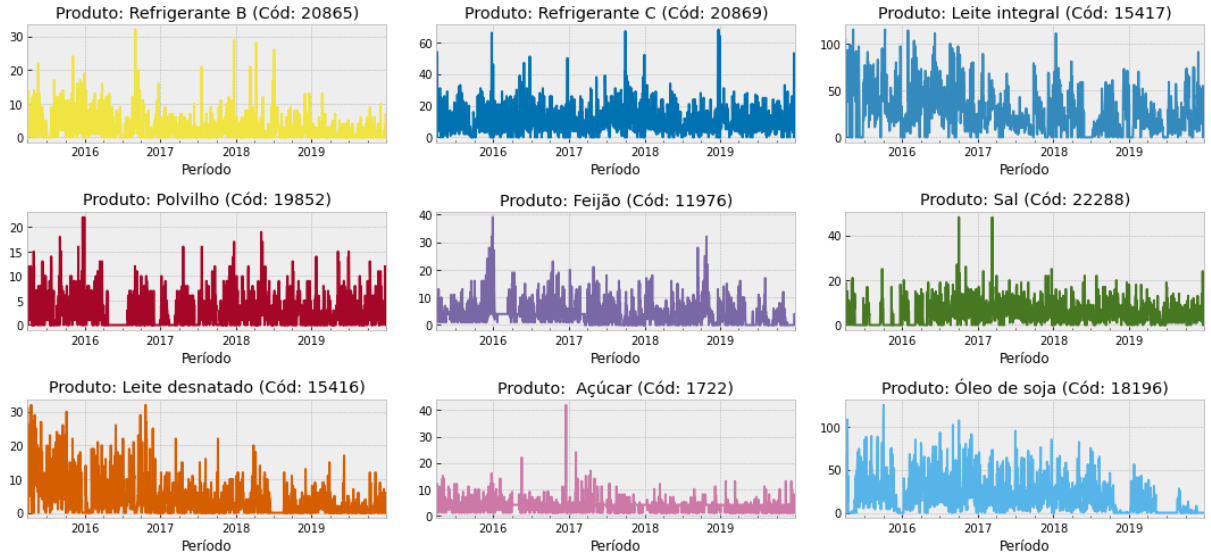
Figura 24 – Amostra da venda de três produtos por dia, mês e ano.



Fonte: Do autor (2021).

As vendas de alguns produtos ficam indisponíveis em curtos períodos, enquanto outros apresentam picos em sua demanda. Diariamente os produtos de códigos 20.865 (Refrigerante B), 19.852 (Polvilho) e 22.288 (Sal) são poucos vendidos, mas por outro lado uma minoria deles é vendida mais que 50 unidades em um dia (FIGURA 25).

Figura 25 – Amostra de venda de nove produtos ao longo do tempo.



Fonte: Do autor (2021).

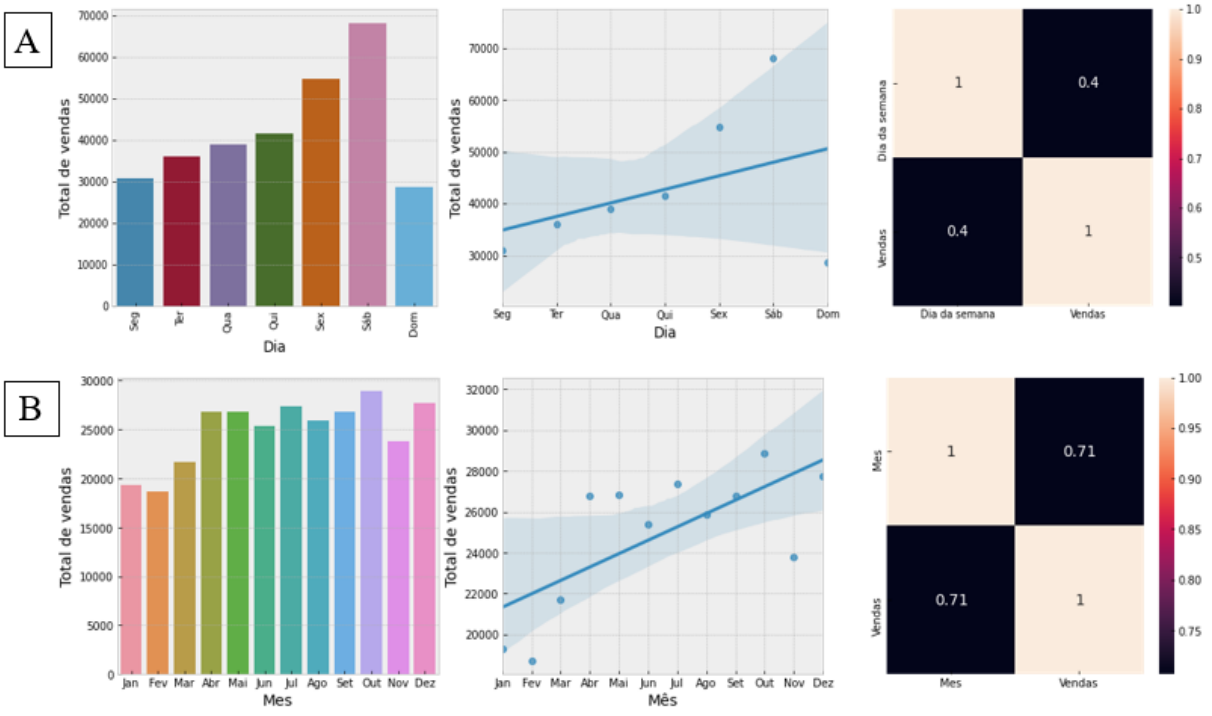
As vendas do dia de sábado para o dia de domingo apresentaram uma queda, esse efeito pode ter sido ocasionado pelo final de semana. A correlação entre os atributos “dia da semana” e “vendas” foi positiva e fraca, indicando pouco aumento na ocorrência das vendas com o passar dos dias (FIGURA 26 A).

A correlação entre os atributos “mês” e “vendas”, foi positiva e forte, indicando que as vendas aumentaram no decorrer dos meses. Entretanto, no mês de outubro notou-se um maior número de vendas em relação aos demais meses (FIGURA 26 B).

Ao correlacionar os atributos “ano” e vendas”, notou-se correlação muito fraca entre ambos, por outro lado, o ano de 2016 foi o que teve maiores registros para vendas (FIGURA 27 A).

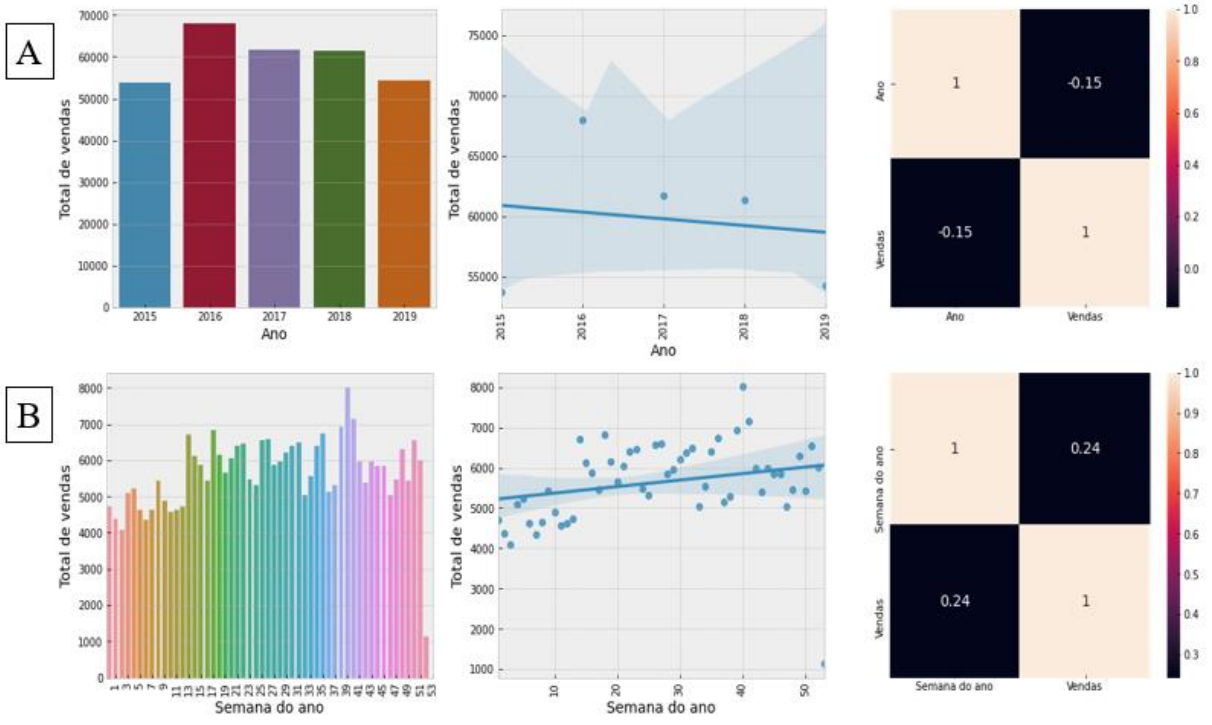
Os atributos “vendas” e “semana do ano” também possuem uma correlação muito fraca. No período da semana 52 para 53, foi registrada uma queda nas vendas, o que pode ter influenciado nesse decaimento é o ano de 2016 o qual foi bissexto e possuiu 53 semanas, sendo o único ano com esse tipo de ocorrência no conjunto de dados. O maior pico de vendas dos produtos ocorreu na semana 39, que corresponde às vendas registradas nos meses de setembro e outubro (FIGURA 27 B).

Figura 26 – Vendas agregadas por dia da semana e mês.



Fonte: Do autor (2021).

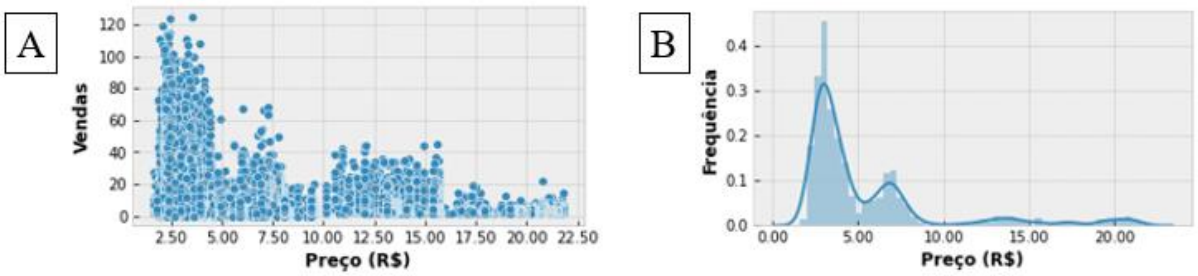
Figura 27 – Vendas agregadas por ano e semana do ano.



Fonte: Do autor (2021).

A demanda dos produtos aumentou à medida que os preços foram diminuindo, conforme o gráfico de dispersão (FIGURA 28 A), esses resultados corroboram com a lei da demanda que pode ser definida de acordo com Vasconcelos e Garcia (2019), que quanto maior o preço há uma queda nas vendas e vice-versa. Além disso, foi observado que a distribuição no gráfico dos preços era concentrada à esquerda do gráfico, como se percebe na Figura 28 B.

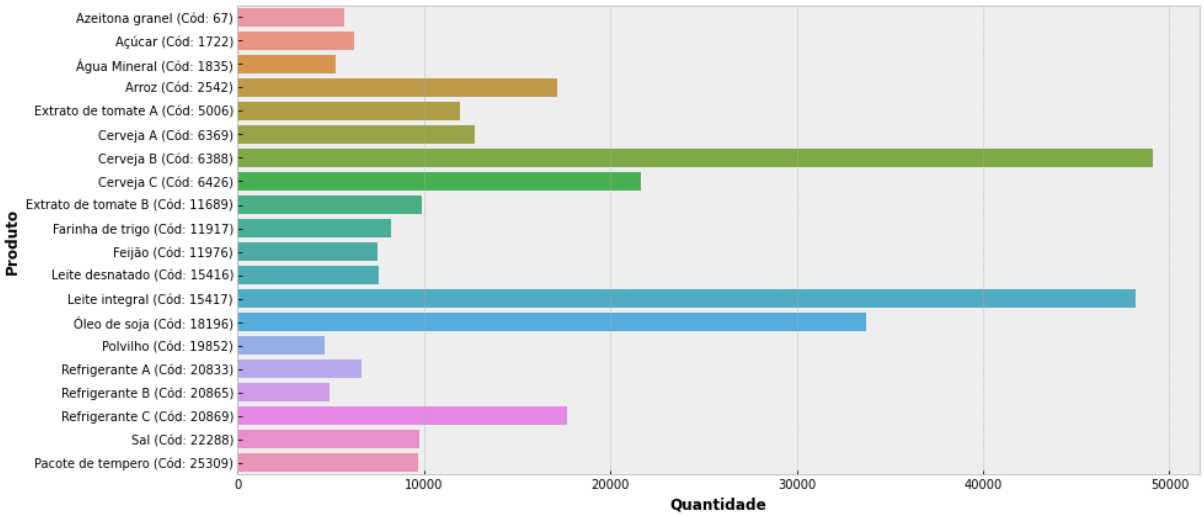
Figura 28 – Relação preço e demanda.



Fonte: Do autor (2021).

Alguns produtos possuíam desempenho de vendas semelhantes durante o período definido, tais como os produtos de código 6.388 (Cerveja B) e 15.417 (Leite integral), os quais foram os mais vendidos concomitantemente durante todo o período. Por outro lado, os produtos com menores registros de venda foram os de códigos 19.852 (Polvilho) e 20.865 (Refrigerante B) (FIGURA 29).

Figura 29 – Demanda por produto.

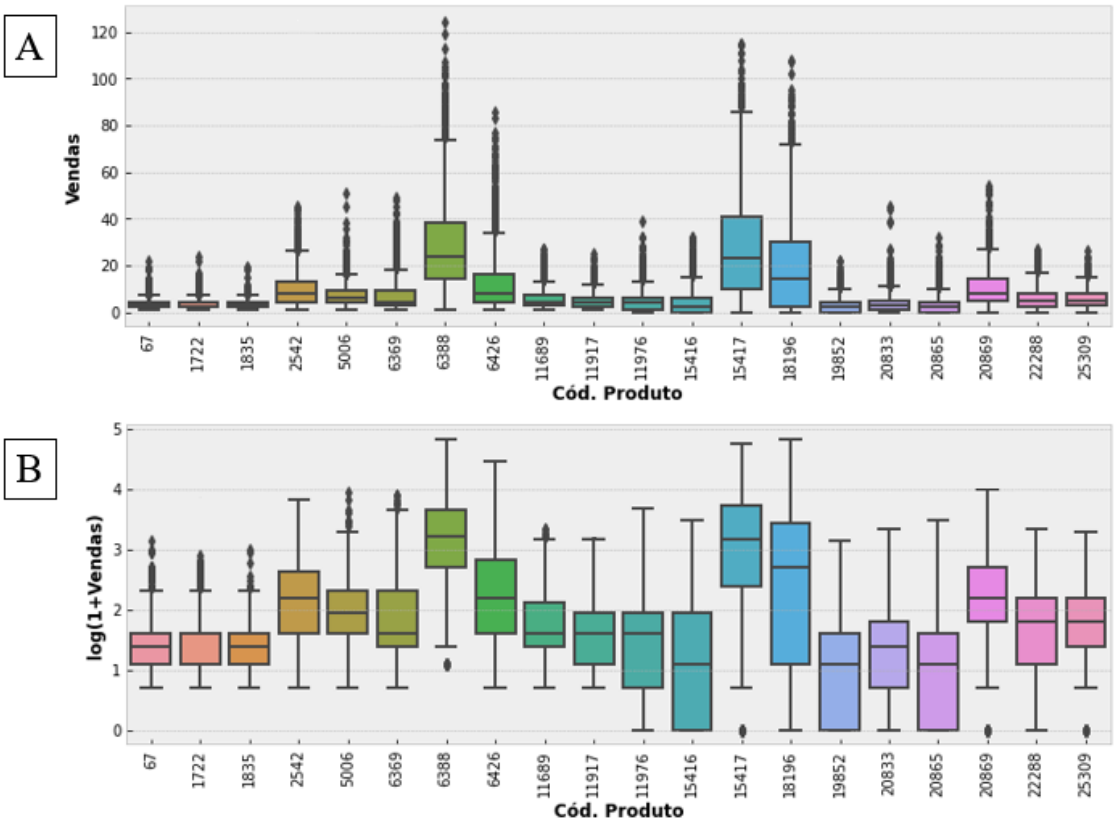


Fonte: Do autor (2021).

No atributo “vendas” observou-se aumentos significativos que comumente ocorrem na rotina do varejo, pois, as vendas são influenciadas pelo comportamento dos consumidores e até mesmo por promoções (FIGURA 30 A) (VASCONCELOS; GARCIA, 2019). Para tratar esses aumentos significativos nos dados, foi realizada a transformação logarítmica dos dados conforme Equação 2.43, em que x representa o atributo “vendas” (FIGURA 30 B).

$$x = \log_{10}(1 + x) \tag{2.43}$$

Figura 30 – *Boxplot* das vendas por produto.



Fonte: Do autor (2021).

A seleção dos atributos mais importantes da série de demandas foi possível através da aplicação do algoritmo de Floresta Aleatória o qual era composto por 1.000 árvores. De acordo com essa seleção dos atributos foi possível plotar um gráfico (FIGURA 31) em que foi definida suas respectivas relevâncias. Neste gráfico, a linha vermelha pontilhada e a azul contínua com marcadores representaram, respectivamente, o grau de importância dentro de um percentual de 95% e a soma cumulativa de importância dos atributos.

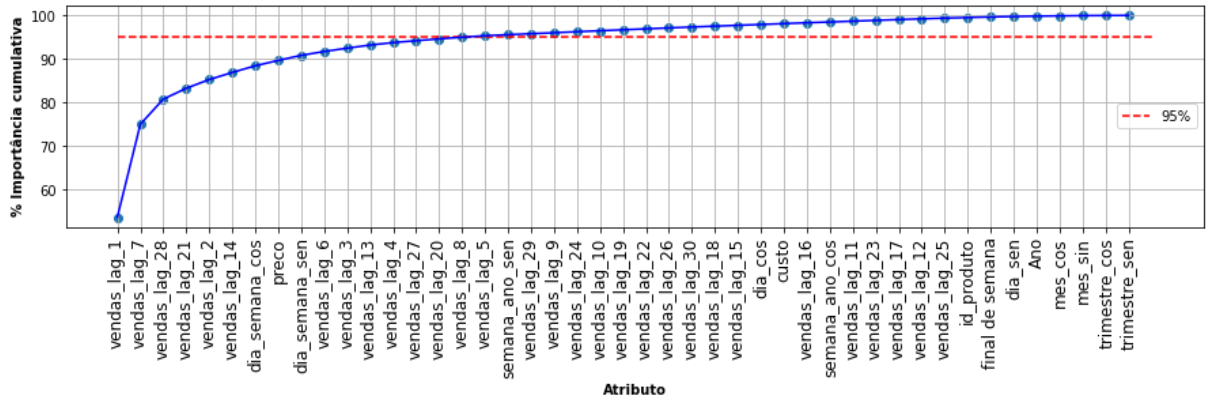
Desta forma os atributos selecionados como entrada para o treinamento dos modelos de aprendizagem de máquina foram: “vendas_lag_1”, “vendas_lag_7”, “vendas_lag_28”, “vendas_lag_21”, “vendas_lag_2”, “vendas_lag_14”, “dia_semana_cos”, “preço”, “dia_semana_sen”, “vendas_lag_6”, “vendas_lag_3”, “vendas_lag_13”, “vendas_lag_4”, “vendas_lag_27”, “vendas_lag_20” e “vendas_lag_8” (FIGURA 31).

Os atributos “vendas_lag_1” e “vendas_lag_7” foram selecionados e ao serem somados representaram um total de aproximadamente 73% de relevância. As defasagens em número de dias do atributo “vendas” são importantes para os produtos que não possuem

tendências, mas que apresentam sazonalidade, pois, as vendas dos dias anteriores tornam-se mais próximas das vendas dos dias mais recentes.

Os atributos “dia_semana_cos” e “dia_semana_sen” apresentaram somados a uma relevância de 3%, esse percentual ocorreu devido às características da sazonalidade presente, e o atributo “preço” foi selecionado e apresentou influência nas vendas.

Figura 31 – Importância dos atributos da série de demanda.

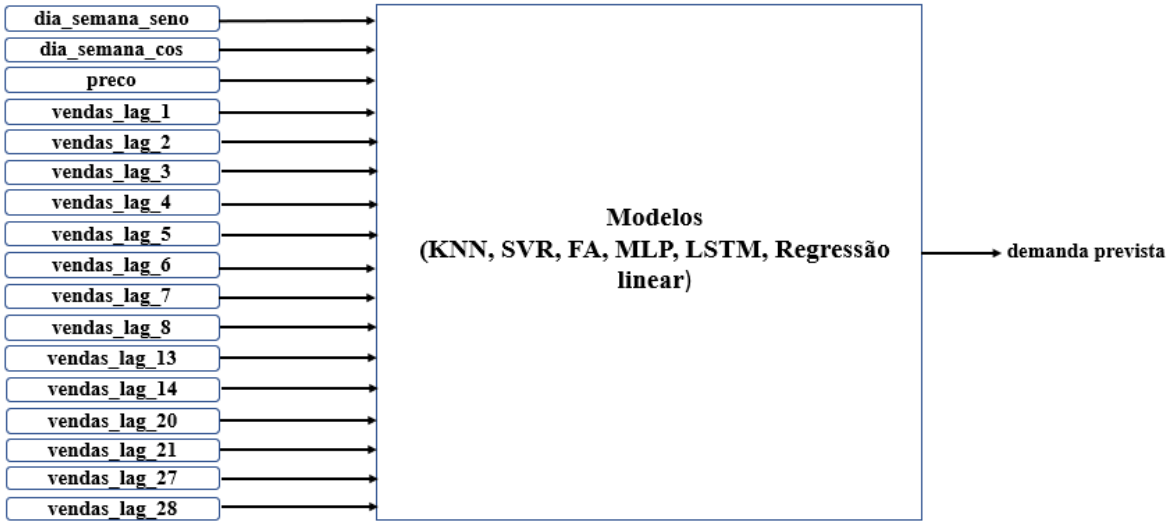


Fonte: Do autor (2021).

4.2.1 Modelos de aprendizado de máquina para previsão de demandas

Para treinamento dos modelos de aprendizado de máquina para previsão de demanda, utilizou-se os atributos mais relevantes selecionados na seção anterior como entrada em forma de vetores para os modelos (FIGURA 32).

Figura 32 – Entradas e saída dos modelos para previsão de demanda.



Fonte: Do autor (2021).

As médias e o desvio padrão das métricas RMSE, RMSLE, MAE e R^2, e o tempo computacional de 13 períodos de validação apresentadas na Tabela 10 são provenientes dos modelos de Floresta aleatória, Regressão de Vetores de Suporte (SVR), K vizinhos mais próximos (KNN), Rede Neural Artificial *Long Short Term Memory* (LSTM), Rede Neural *Perceptron* multicamadas (MLP) e Regressão linear.

O modelo de Floresta aleatória obteve melhores avaliações para todas as métricas e os valores calculados para o RMSE, RMSLE, MAE e R^2 foram de 55,5817, 0,6349, 3,94 e 0,57 respectivamente.

Os modelos SVR, KNN e Rede Neural MLP apresentaram valores próximos para a métrica do RMSE. Entretanto, notou-se que em dois períodos de validação da Rede Neural MLP os valores obtidos eram discrepantes em relação à média calculada (FIGURA 33).

Em relação à Regressão Linear os resultados das métricas calculadas foram inferiores de forma geral quando comparados aos demais modelos, entretanto enfatiza-se a importância da utilização desse modelo como uma linha de base para comparação com os demais modelos.

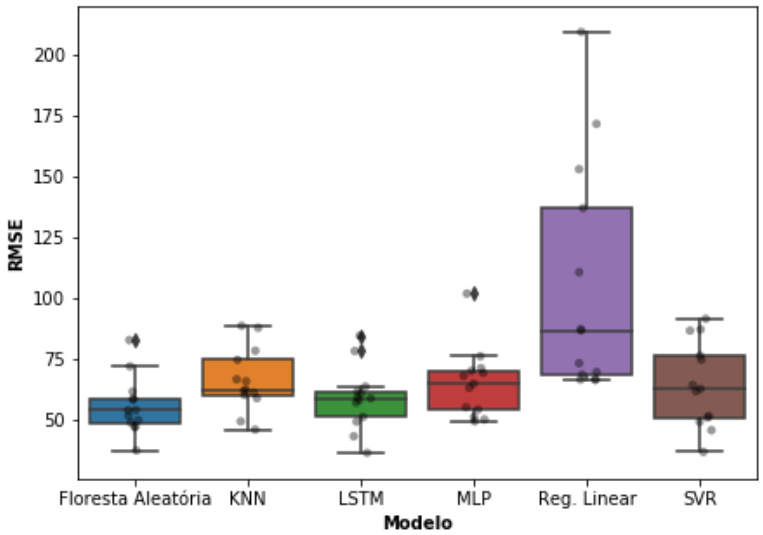
Os modelos que demandaram maior tempo computacional para treinamento foram o LSTM e MLP, com um total de 04h:19m:48s e 02h:45m:04s respectivamente. Por outro lado, os modelos de Regressão linear e SVR demandaram menor tempo computacional.

Em relação ao tempo computacional demandado para o treinamento dos modelos, deve-se avaliar o objetivo da empresa, pois, modelos mais rápidos com desempenho abaixo do desejável podem ser preferíveis aos mais lentos (NIKOLOPOULOS; PETROPOULOS, 2018; SPILIOTIS et al., 2020).

Tabela 10 – Resultados para série temporal de demanda.

Modelo	RMSE médio	Desvio padrão RMSE	RMSLE médio	Desvio padrão RMSLE	MAE médio	Desvio padrão MAE	R^2 médio	Desvio padrão R^2	Tempo computacional
KNN	66,2549	13,0153	0,72900	0,0443	4,46	0,46	0,49	0,08	00:07:52
LSTM	58,6022	12,8316	0,68081	0,4437	4,20	0,47	0,54	0,08	04:19:48
MLP	64,9447	14,3027	0,64954	0,0429	4,17	0,42	0,51	0,09	02:45:04
Floresta Aleatória	**55,5817**	11,6788	**0,63493**	**0,0362**	**3,94**	**0,40**	**0,57**	**0,07**	**00:30:28**
SVR	64,5233	17,5645	0,64346	0,0383	4,11	0,47	0,39	0,09	00:04:01
Regressão Linear	105,1130	47,7981	0,72342	0,0410	4,68	0,50	0,24	0,23	00:00:49

Figura 33 – *Boxplot* do RMSE da previsão de demanda.

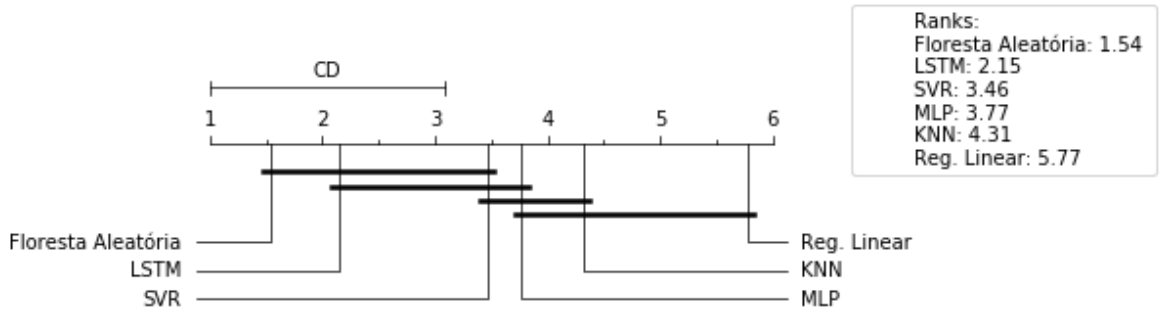


Fonte: Do autor (2021).

As médias calculadas para o RMSE dos 13 períodos de validação dos modelos foram significativas conforme teste de *Friedman* (x_r^2 = 42,846 e p = 3,97010345E-8). No teste de *Nemeneyi* aplicado posteriormente para todos os modelos, a Floresta aleatória apresentou diferenças significativas quando comparada com os modelos KNN, MLP e Regressão linear. Já no modelo LSTM o teste de *Nemenyi* também apresentou diferença significativa quando comparado com os modelos KNN e Regressão linear, por outro lado, em relação a Floresta aleatória, SVR e MLP não houve diferença significativa (FIGURA 34).

E por fim, no modelo SVR o teste de *Nemenyi* demonstrou diferença significativa quando comparado com o modelo de Regressão linear, e os demais modelos KNN, MLP e Regressão linear não apresentaram diferença significativa entre si (FIGURA 34).

Figura 34 – Comparação dos modelos da série temporal de demanda através do teste de *Friedman* e Pós-Teste *Nemenyi*, com uma distância crítica (CD) de 2.09.



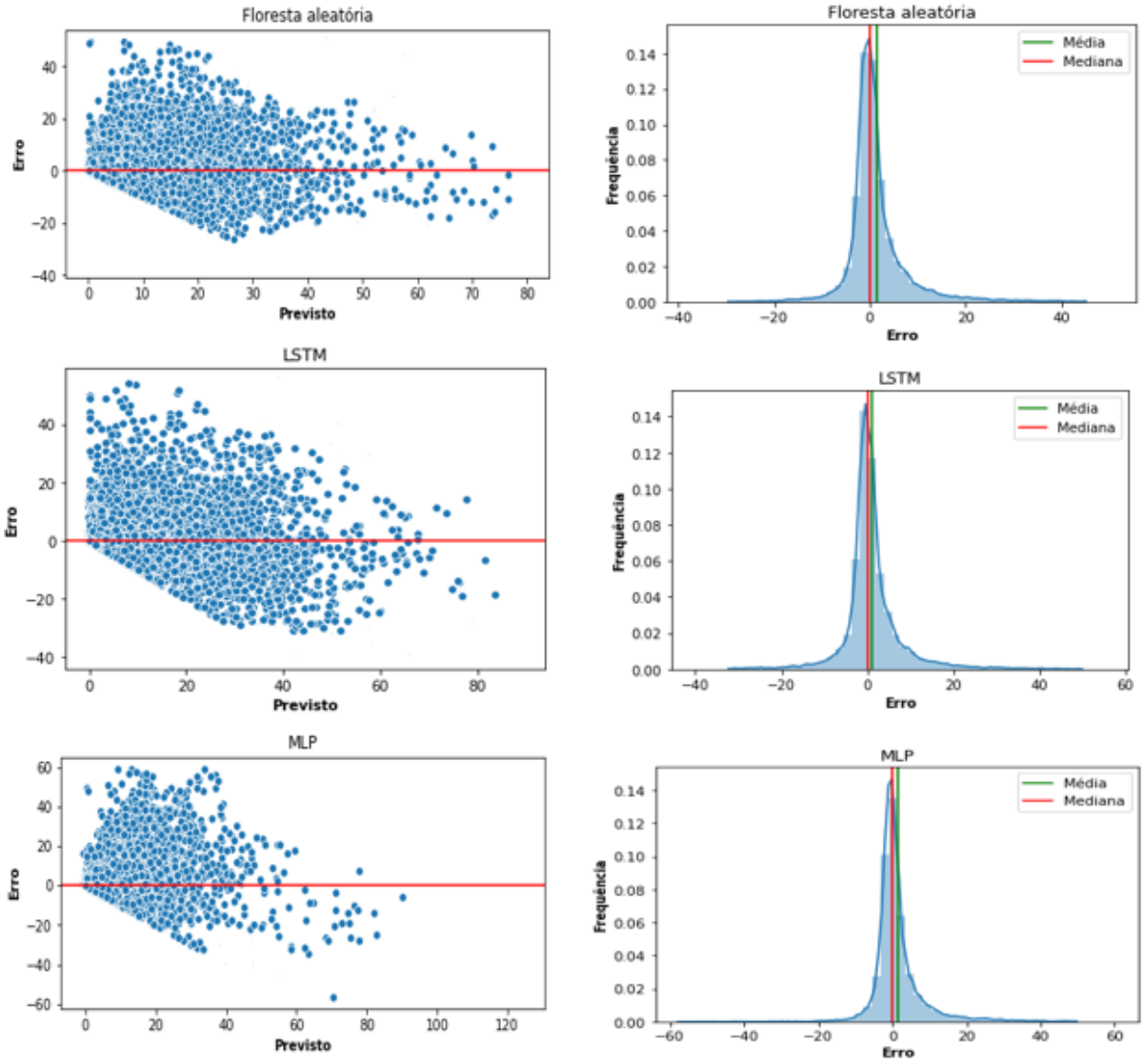
Fonte: Do autor (2021).

Os erros dos modelos de aprendizado de máquina possuem uma distribuição assimétrica. Observou-se que o maior erro foi obtido pelo modelo de Regressão linear, além disso, todas as previsões acima de 80 foram superestimadas pelo modelo (Figuras 35 e 36).

As previsões de demandas com valores entre 0 e 20 apresentaram os maiores erros para os modelos de Floresta aleatória, LSTM, SVR e KNN, contudo, os erros tendem a ser menos dispersos e menores para previsões maiores que 60 (FIGURAS 35 e 36).

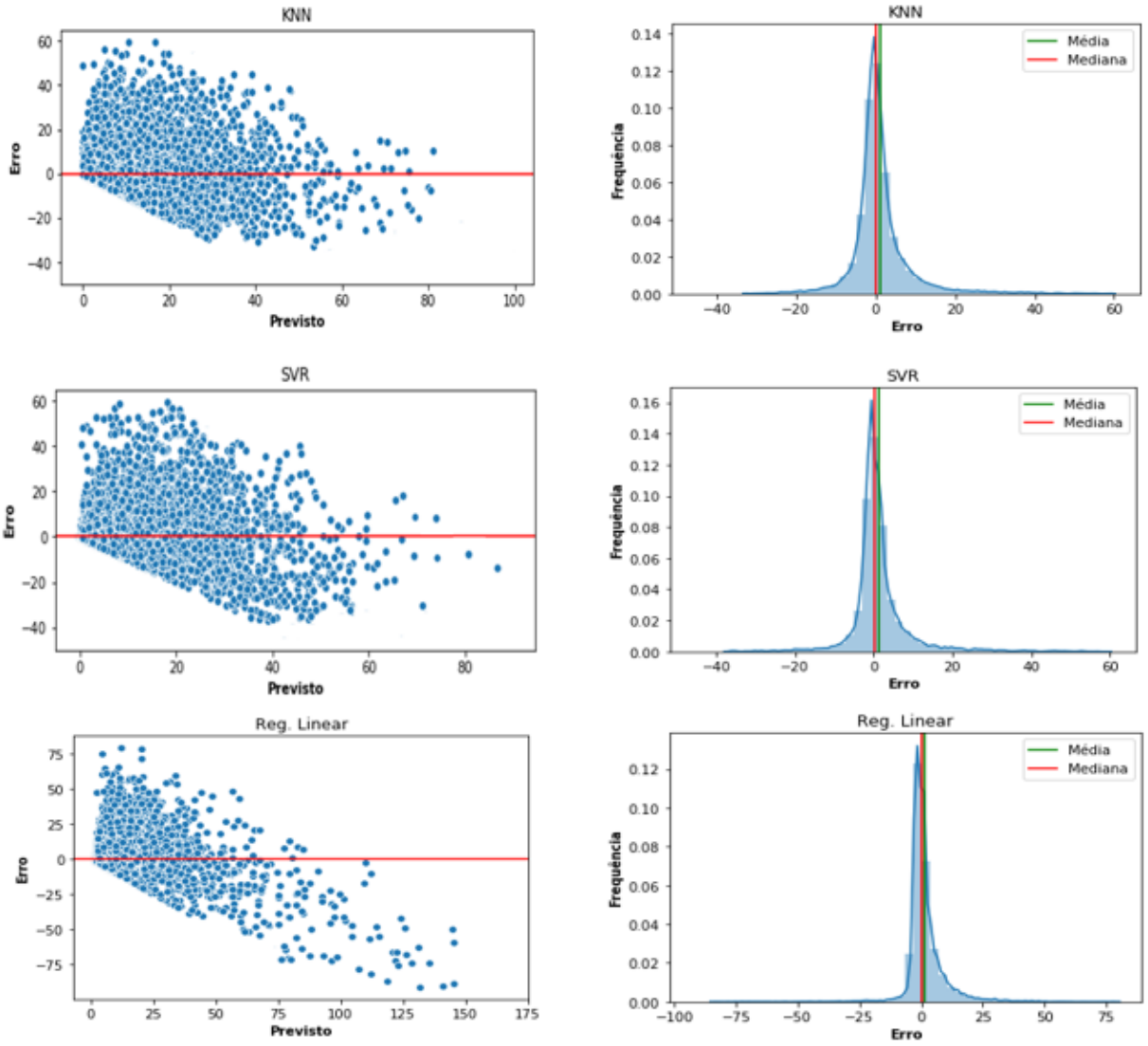
O modelo MLP apresentou maiores erros calculados para as previsões com valores entre 0 e 20 e 30 a 40. Além disso, as previsões acima de 60 foram maiores em relação à demanda observada (FIGURAS 35 e 36).

Figura 35 – Resíduos dos modelos de Floresta aleatória, LSTM e MLP na previsão de demandas.



Fonte: Do autor (2021).

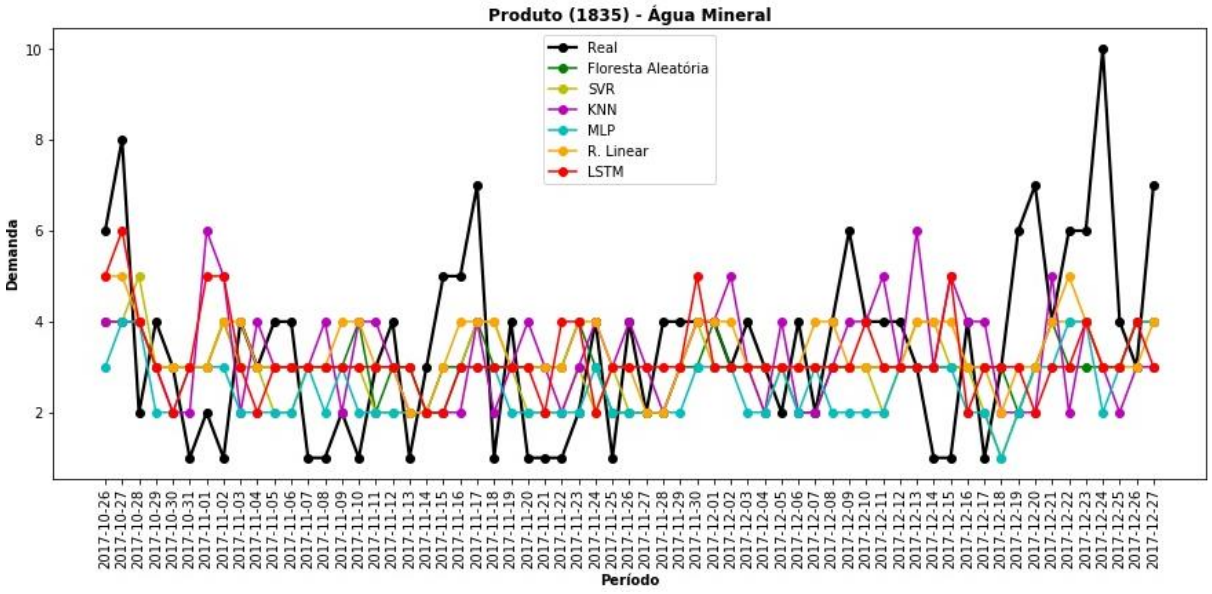
Figura 36 – Resíduos dos modelos de KNN, SVR e Regressão linear na previsão de demandas.



Fonte: Do autor (2021).

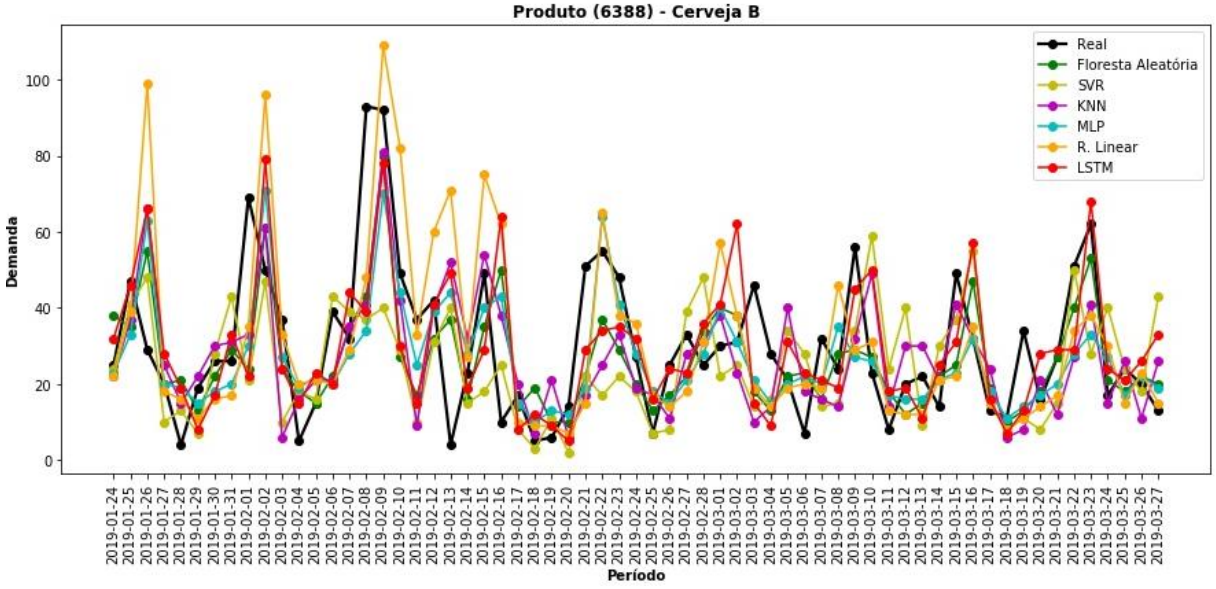
Observou-se que em dias que não foram registrados a saída de produto pela ausência de vendas, os modelos de aprendizagem de máquina não conseguiram prever corretamente. Além disso, foram observadas ocorrências de previsão de demanda semelhantes ao produto de código 1.835 (Água mineral), em que os valores das previsões da demanda não se aproximaram dos valores esperados quando havia picos em sua demanda. Para os produtos de códigos 6.388 (Cerveja B) e 20.869 (Refrigerante C) as previsões de demanda se aproximaram dos valores esperados até mesmo quando acompanhados de picos em suas demandas (FIGURAS 37 a 39).

Figura 37 – Previsão da demanda x valor esperado para o produto 1835.



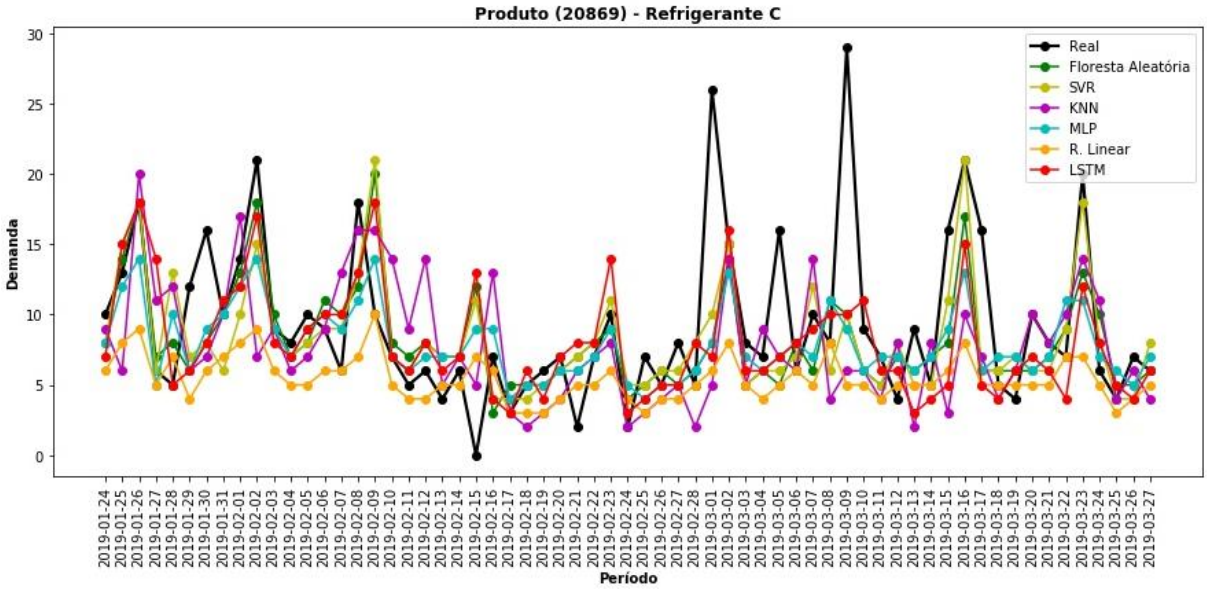
Fonte: Do autor (2021).

Figura 38 – Previsão da demanda x valor esperado para o produto de código 6388.



Fonte: Do autor (2021).

Figura 39 – Previsão da demanda x valor esperado para o produto código 20869.

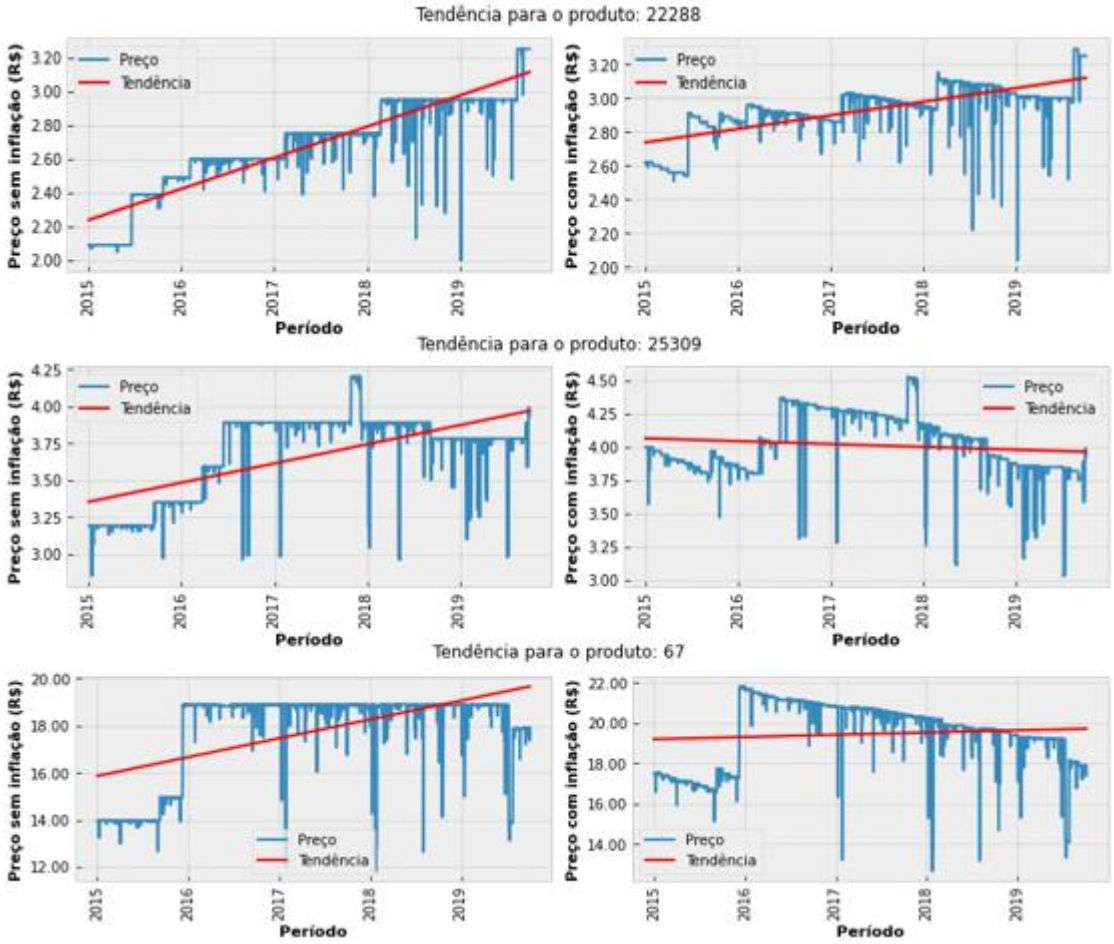


Fonte: Do autor (2021).

4.3 Análise e preparação dos dados para série temporal de preços

O comportamento dos preços de três produtos (22.288 – Sal, 25.309 – Pacote de tempero e 67 – Azeitona), avaliados diariamente mostrou uma linha de tendência crescente ao longo do tempo, porém, após inflacionar o atributo “preço” uma redução nessa linha de tendência foi observada. Desta forma, é possível inferir que nem toda linha de tendência dos preços pode ser explicada pela inflação (FIGURA 40).

Figura 40 – Tendência do preço ao longo do tempo.

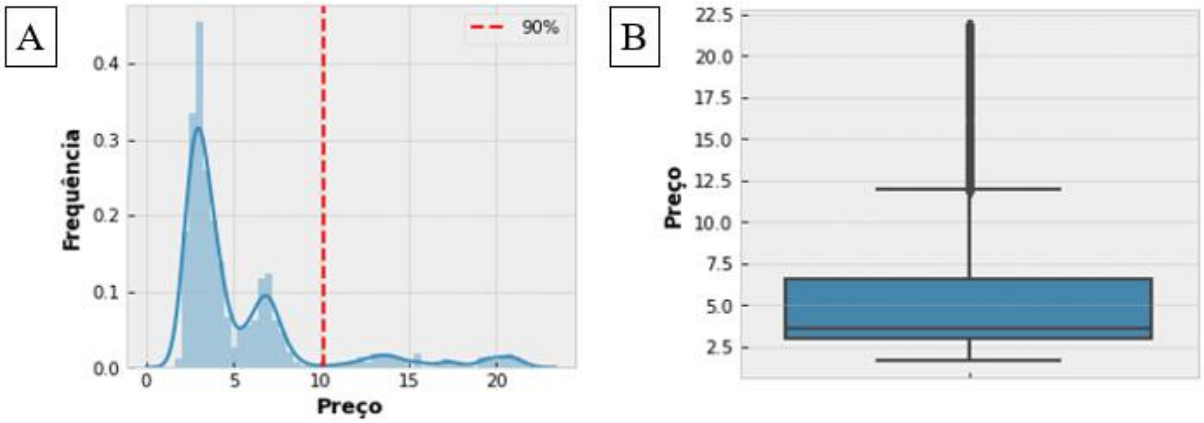


Fonte: Do autor (2021).

Em relação aos registros do atributo “preço”, 90% deles apresentaram um valor inferior a R$ 10,00 (FIGURA 41 A). Além disso, valores discrepantes foram observados em relação à média do atributo “preço” devido a diferença de preços entre produtos (Figura 41 B).

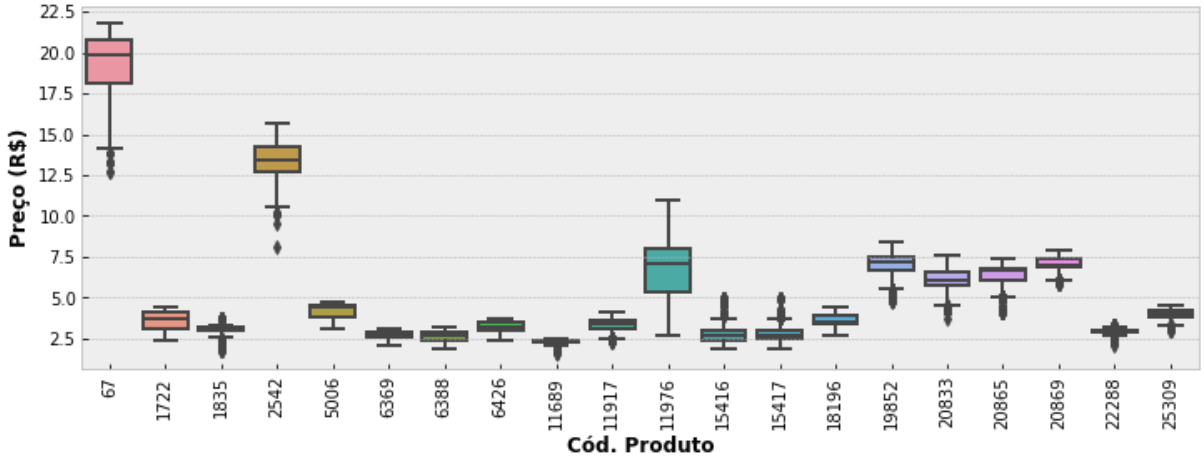
O atributo “preço” dos produtos com maior variabilidade foram os de códigos 67 (azeitona) e 11.976 (feijão), porém esse mesmo atributo apresentou valores discrepantes para alguns produtos como exemplo os de códigos 2.542 (Arroz), 11.917 (Farinha de trigo), 15.416 (Leite desnatado) entre outros (FIGURA 42). Apesar disso, os valores discrepantes estão bem próximos do limite inferior e superior do *boxplot*, e devido ao atributo “preço” ser suscetível a variações diárias como descontos e acréscimos, por exemplo, esses valores foram mantidos no conjunto de dados.

Figura 41 – Distribuição e *boxplot* geral do preço.



Fonte: Do autor (2021).

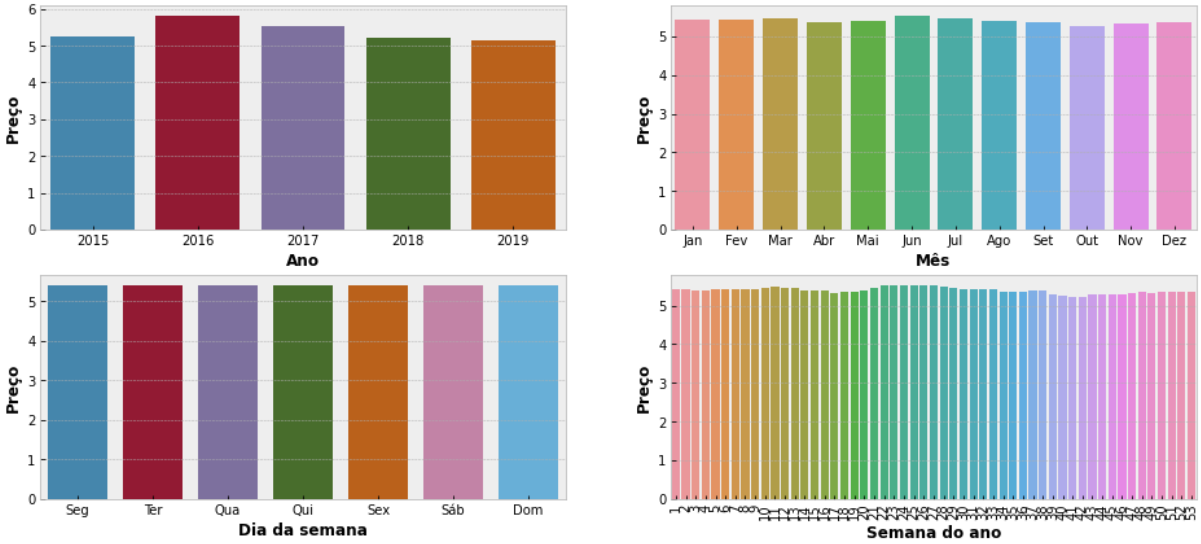
Figura 42 – Variações de preços por produto.



Fonte: Do autor (2021).

O atributo “preço” quando comparado com os atributos sazonais como “ano”, “mês”, “dia da semana” e “semana do ano”, apresentaram uma distribuição uniforme no decorrer do tempo, isso implica que os atributos sazonais podem não ser tão relevantes para os modelos de aprendizado de máquina para previsão de preços (FIGURA 43).

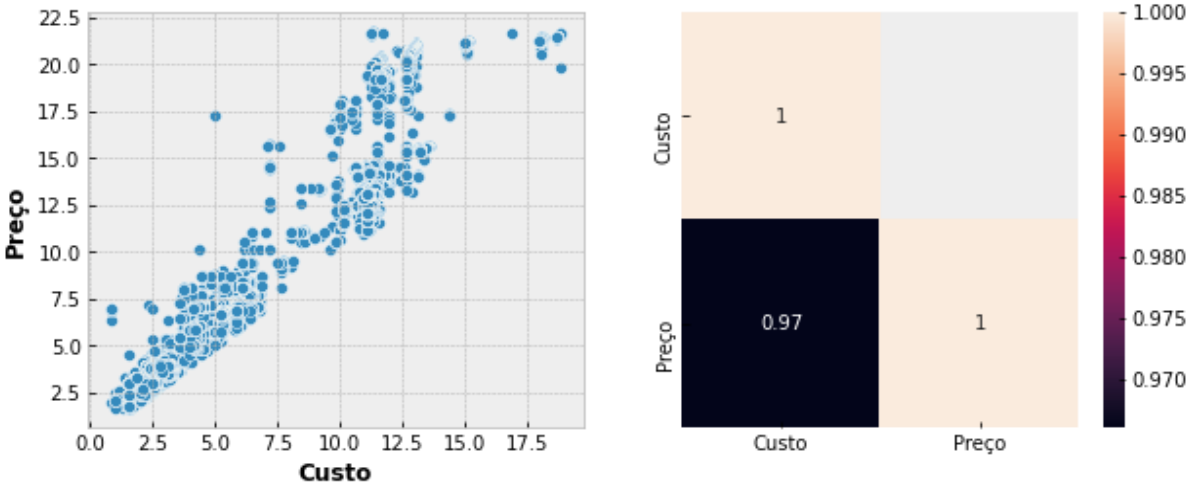
Figura 43 – Preço por ano, mês, dia da semana e semana do ano.



Fonte: Do autor (2021).

O atributo “custo” correlaciona-se com o atributo “preço” de maneira positiva e muito forte, ou seja, são diretamente proporcionais. Desta forma, o custo pago ao fornecedor pelo varejista para a aquisição do produto tem forte influência no preço de venda (FIGURA 44).

Figura 44 – Relação custo x preço.



Fonte: Do autor (2021).

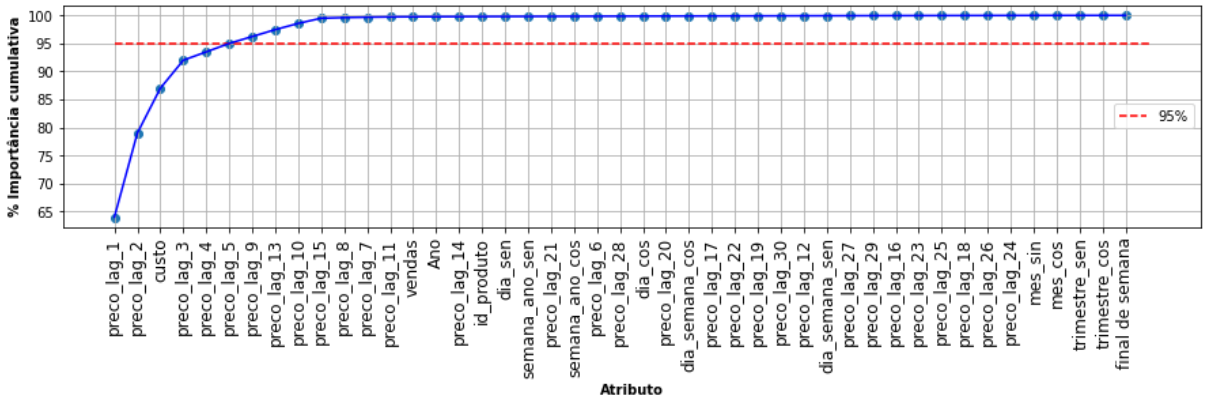
A seleção dos atributos mais importantes da série de preços foi possível através da aplicação do algoritmo de Floresta Aleatória, composto por 1.000 árvores. De acordo com essa seleção dos atributos, um gráfico foi gerado e assim definiram-se as relevâncias de cada um respectivamente (FIGURA 45). A linha pontilhada vermelha presente no gráfico da Figura

45 refere-se ao grau de importância dentro de um percentual de 95% e a linha azul continua com marcadores representa a importância cumulativa dos atributos.

Os atributos que foram selecionados como entrada para o treinamento ou ajuste dos modelos de aprendizagem de máquina foram: “preco_lag_1”, “preco_lag_2”, “preco_lag_3”, “preco_lag_4”, “preço_lag_5” e “custo”. Dentre eles o atributo mais relevante foi o “preco_lag_1” com 64% de relevância, seguido pelos atributos “preco_lag_2”, “custo”, “preco_lag_3”, “preco_lag_4” e “preço_lag_5” (FIGURA 45).

Devido a esses atributos acima descritos serem relacionados com os preços dos dias anteriores, eles são importantes para os modelos de aprendizagem de máquina, pois os preços não tendem a alterar de um dia para o outro de forma acentuada. Além disso, o atributo “custo” também foi relevante para os modelos descritos, pois, influenciaram o atributo “preço”.

Figura 45 – Variáveis relevantes para série temporal de preços.

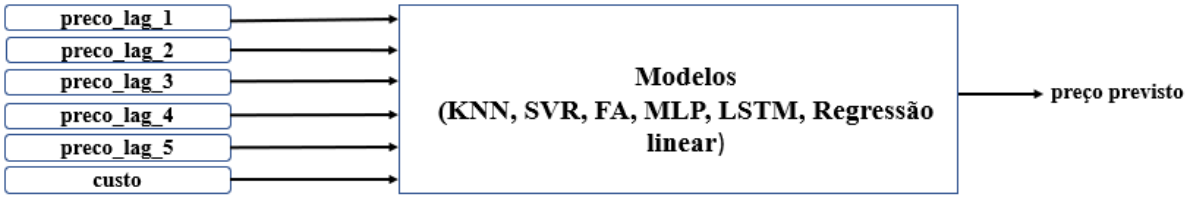


Fonte: Do autor (2021).

4.3.1 Modelos de aprendizado de máquina para previsão de preços

Para a previsão dos preços, os atributos de entrada para treinamento dos modelos de aprendizado de máquina foram: “preco_lag_1”, “preco_lag_2”, “preco_lag_3”, “preco_lag_4”, “preco_lag_5” e “custo” (FIGURA 46).

Figura 46 – Entradas e saída dos modelos para previsão de preço.



Fonte: Do autor (2021).

O modelo da Floresta aleatória apresentou melhor desempenho na previsão dos preços em todas as métricas aplicadas e os valores obtidos para as métricas foram respectivamente: RMSE (0,073930), RMSLE (0,0360), MAE (0,11) e R^2 (0,9981) (TABELA 11).

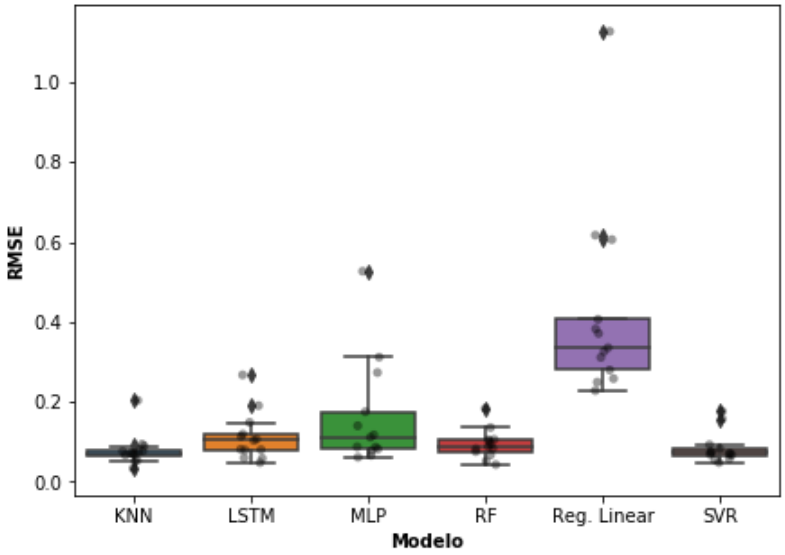
Os modelos LSTM e KNN apresentaram valores calculados para o RMSE semelhantes e o mesmo valor para o MAE. No modelo MLP, as métricas RMSE e MAE quando comparadas aos demais modelos com exceção da Regressão linear apresentaram valores altos. Desta forma, temos que o modelo das Redes Neurais Artificiais MLP possui variedades de hiper parâmetros (número de neurônios e camadas ocultas, funções de ativação dos neurônios, taxa de aprendizagem, algoritmo de otimização, regularização, dentre outros) que implicam diretamente nos resultados. A Regressão linear entre os modelos escolhidos foi a que apresentou um resultado inferior, isso pode ter ocorrido pelo fato da linearidade do modelo (FIGURA 47) (TABELA 11).

Os modelos que exigiram maior tempo computacional para treinamento ou ajuste foram os das Redes Neurais Artificiais, os tempos gastos foram respectivamente de 03h:44m:12s e 02h:26m:14s para a LSTM e MLP. Por outro lado, os modelos de Regressão linear e SVR foram os que exigiram menor custo computacional (TABELA 11).

Tabela 11 – Resultados para série temporal de preços.

Modelo	RMSE	Desvio padrão RMSE	RMSLE médio	Desvio padrão RMSLE	MAE	Desvio padrão MAE	R^2 Médio	Desvio padrão R^2	Tempo computacional
KNN	0,093207	0,038412	0,0407	0,0061	0,13	0,02	0,9977	0,0013	00:06:39
LSTM	0,091102	0,033661	0,0361	0,0029	0,13	0,02	0,9977	0,0010	03:44:12
MLP	0,256311	0,134490	0,0435	0,0090	0,21	0,07	0,9976	0,0010	02:26:14
Floresta Aleatória	**0,073930**	0,040559	**0,0360**	**0,0043**	**0,11**	0,02	**0,9981**	0,0012	00:19:01
SVR	0,083264	0,039295	0,0378	0,0037	0,12	0,02	0,9952	0,0012	00:03:02
Regressão Linear	0,408372	0,226071	0,0837	0,0072	0,43	0,09	0,9756	0,0011	00:00:27

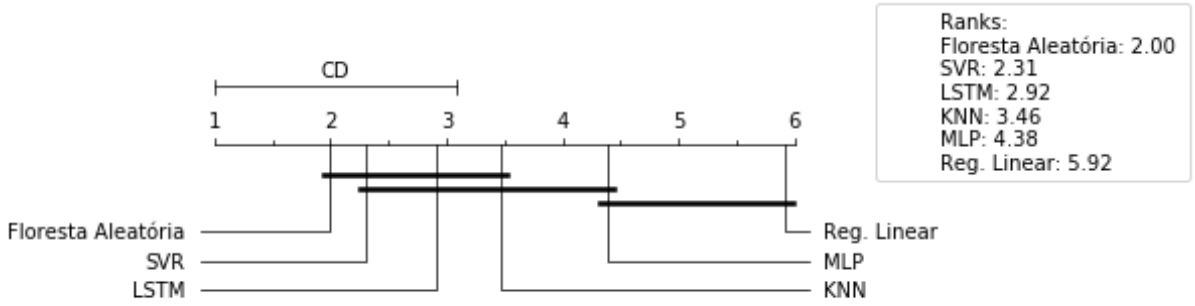
Figura 47 – *Boxplot* do RMSE das previsões de preços.



Fonte: Do autor (2021).

As médias do RMSE calculadas a partir dos 13 períodos de validação dos modelos de aprendizagem de máquina foram significativas de acordo com o teste de *Friedman* ($x_r{}^2$ = 39,593 e p = 1.8035464926002077^{-7}). O teste de *Nemenyi* foi aplicado após o teste de *Friedman* para todos os modelos e a Floresta aleatória apresentou diferença significativa quando comparada com os modelos de Regressão linear e MLP. Para os modelos SVR e LSTM também houve diferenças significativas quando comparados com a Regressão linear (FIGURA 48).

Figura 48 – Comparação dos modelos da série temporal de preços através do teste de *Friedman* e Pós-Teste *Nemenyi*, com uma distância crítica (CD) de 2.09.



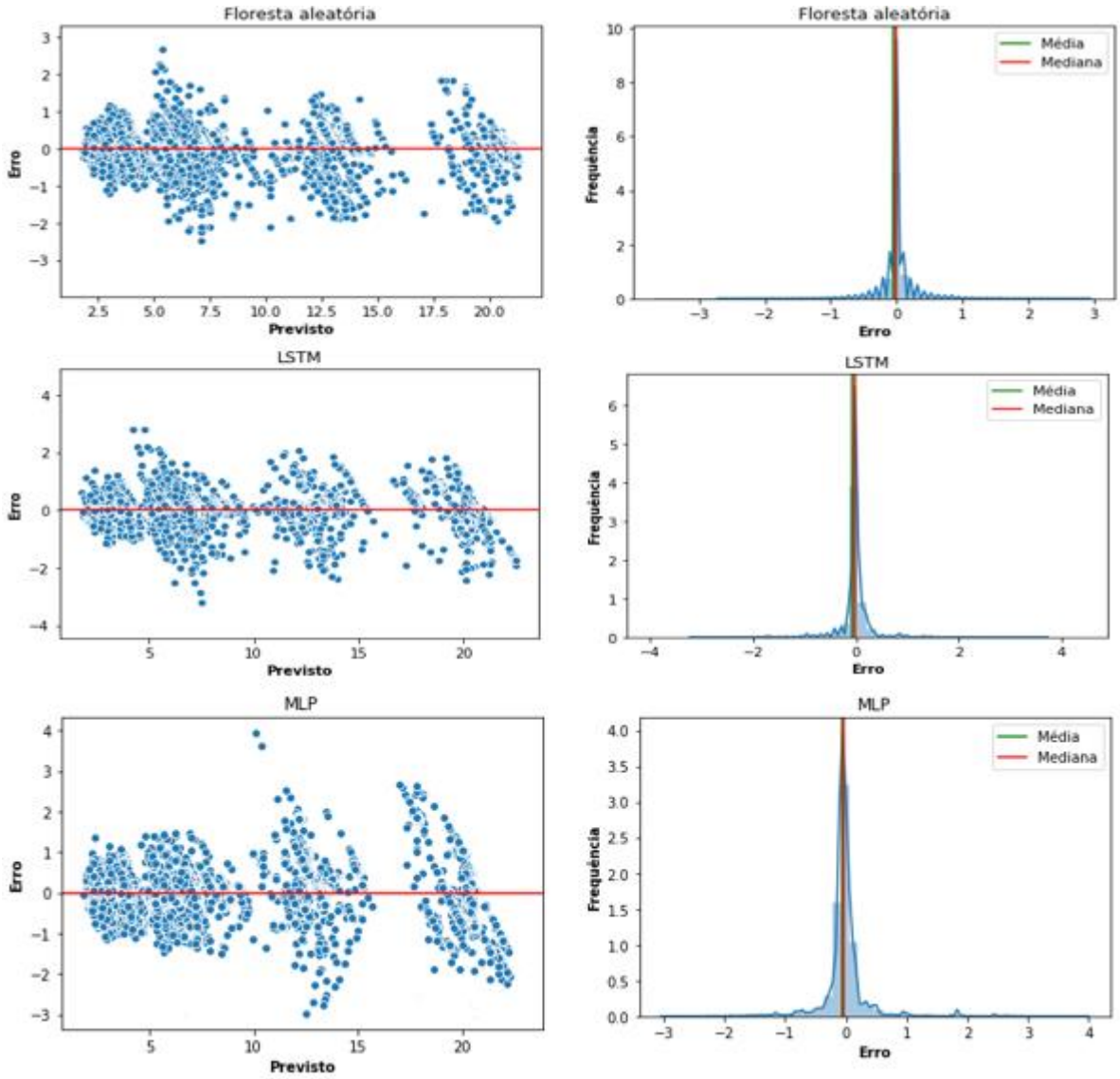
Fonte: Do autor (2021).

Os menores erros observados foram nas previsões entre R$ 2,00 e R$ 5,00 para todos os modelos de aprendizado de máquina. O maior erro para o modelo de Regressão linear

encontrado estava entre as previsões de R$ 14,00 a R$ 16,00. Além disso, o modelo de Regressão linear superestimou a maior parte das previsões acima de R$ 20,00 (FIGURAS 49 e 50).

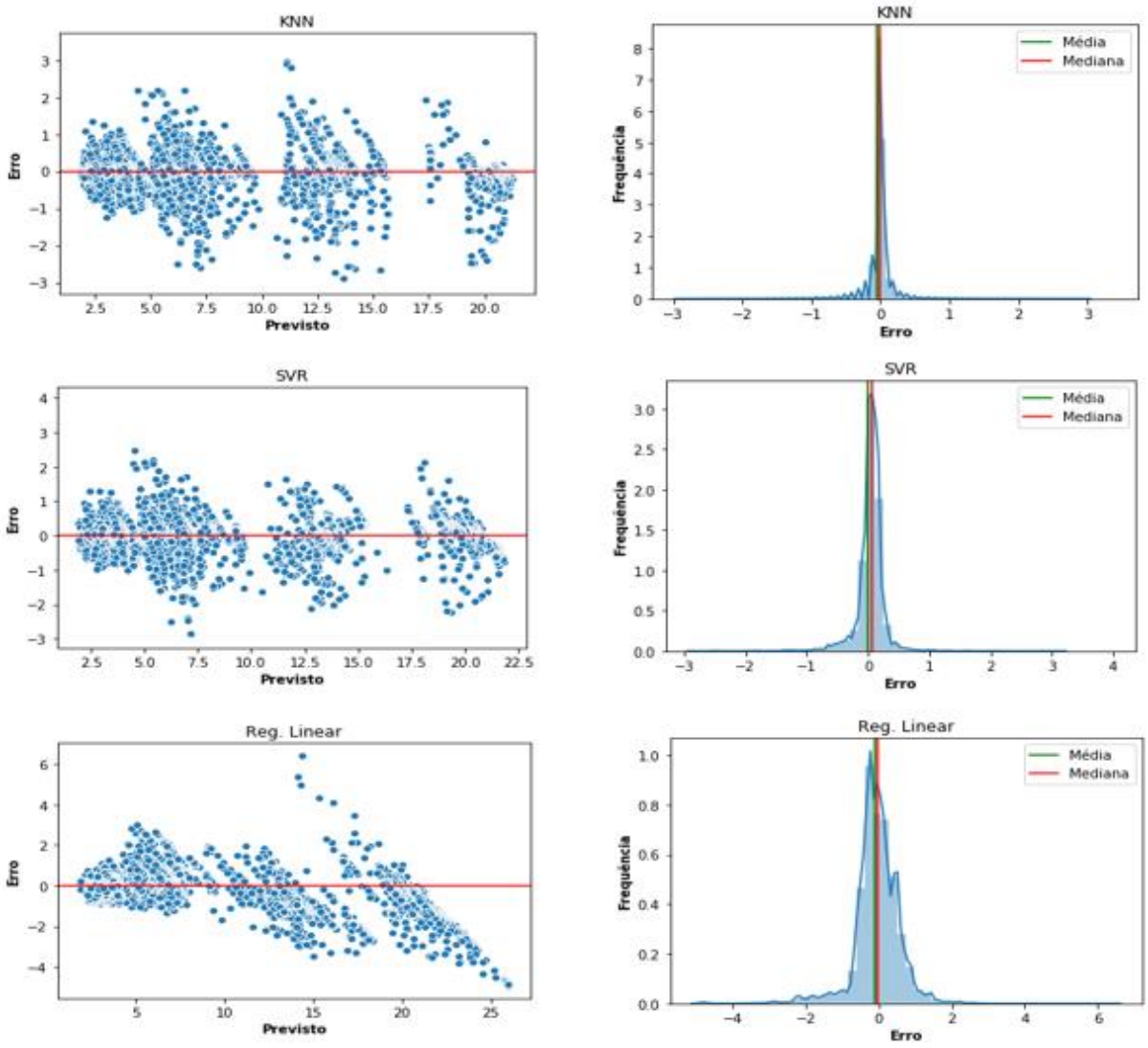
Os modelos de Floresta aleatória, LSTM e SVR, apresentaram dispersão e erros maiores para as previsões entre R$ 5,00 e R$ 7,50 e o modelo MLP demonstrou variabilidade e erros menores, neste intervalo de previsões. Entretanto, houve maior variabilidade de erros para previsões entre R$ 10,00 e R$ 15,00 e de R$ 18,00 a R$ 21,00. O modelo KNN superestimou a maior parte das previsões acima de R$ 18,00. As distribuições dos erros de todos os modelos foram assimétricas (FIGURAS 49 e 50).

Figura 49 – Resíduos dos modelos de Floresta aleatória, LSTM e MLP na previsão de preços.



Fonte: Do autor (2021).

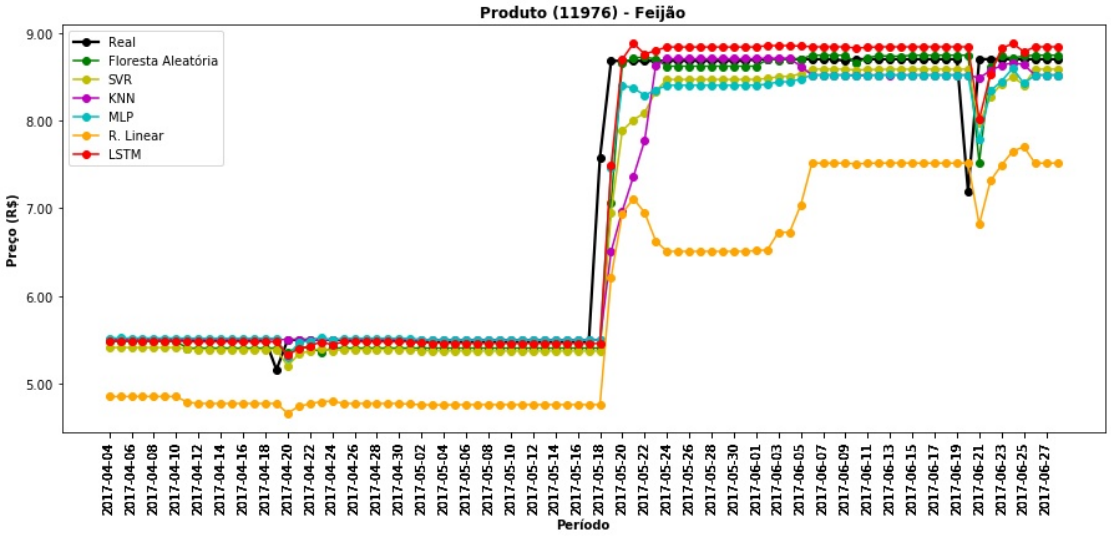
Figura 50 – Resíduos dos modelos de KNN, SVR e Regressão linear na previsão de preços.



Fonte: Do autor (2021).

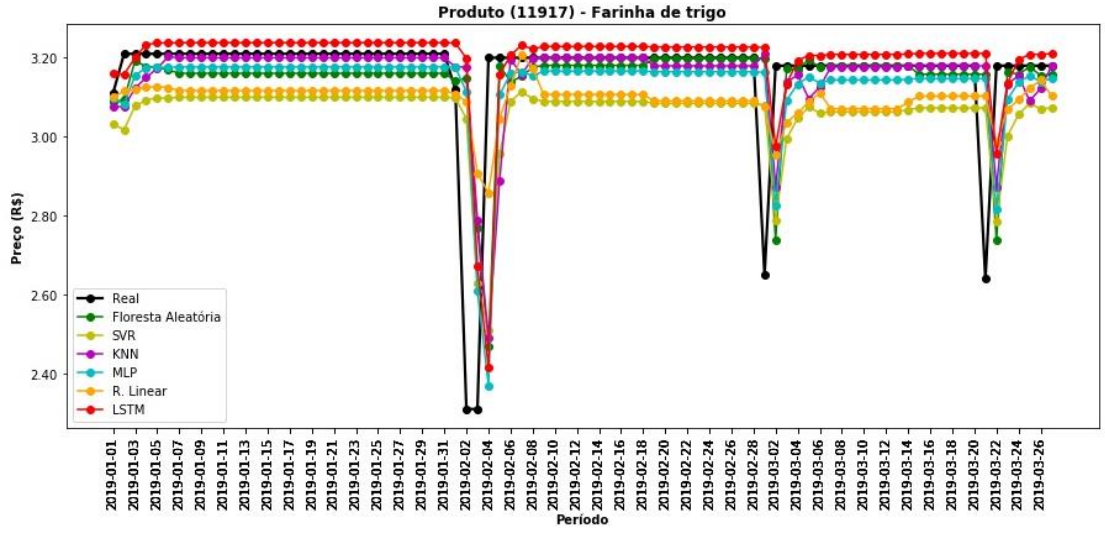
Nas previsões de preços dos produtos de códigos 11.976 (Feijão), 11.917 (Farinha de trigo) e 2.542 (Arroz), foram observados valores bem próximos do esperado além de seguirem a mesma tendência. As previsões dos produtos de códigos 11.917 (Farinha de trigo) e 2.542 (Arroz) se aproximaram dos valores esperados, como podemos observar nos picos e nos vales. Para os produtos de códigos 11.976 (Feijão) e 2.542 (Arroz) as previsões do modelo de Regressão linear se mostraram distantes do valor esperado (FIGURAS 51 a 53).

Figura 51 – Previsão do preço x valor esperado para o produto de código 11976.



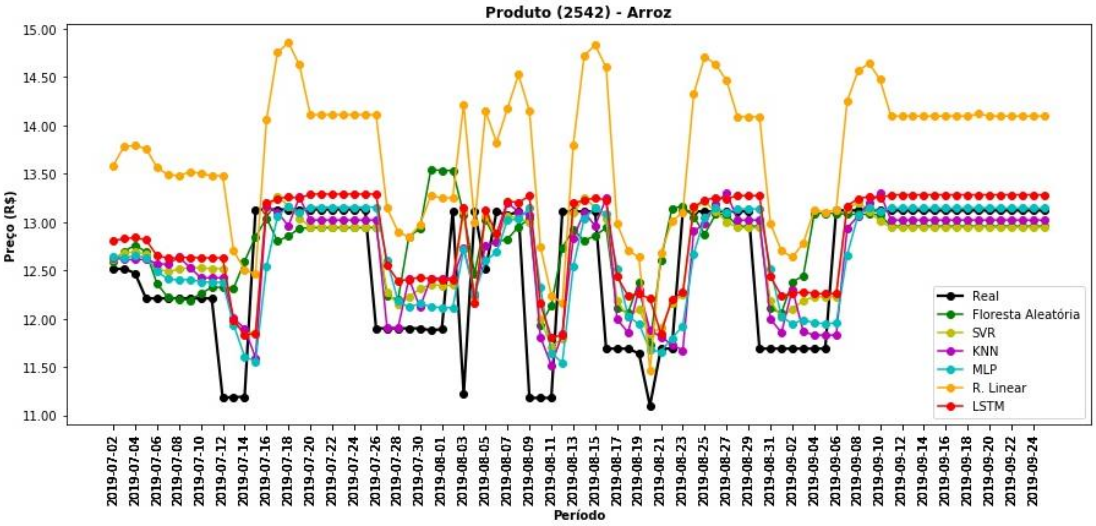
Fonte: Do autor (2021).

Figura 52 – Previsão do preço x valor esperado para o produto de código 11917.



Fonte: Do autor (2021).

Figura 53 – Previsão do preço x valor esperado para o produto de código 2542.



Fonte: Do autor (2021).

5 CONCLUSÃO

As técnicas de aprendizado de máquina no varejo podem contribuir com a compra de estoque de produtos, ou seja, será possível saber quando os varejistas terão que estocar mais produtos devido à elevada saída deles registradas. Além disso, contribui também na tomada de decisão pelos gestores, pois, possibilitam previsões para horizontes de longo prazo.

Neste trabalho foram aplicadas técnicas de aprendizado de máquina tais como: Floresta aleatória, Rede Neural Artificial *Long Short Term Memory*, Rede Neural Artificial MLP, Máquinas de vetores de suporte, K vizinhos mais próximos e Regressão linear para realizar previsões de preços e demandas de produtos do varejo. Para isso, foram comparados os resultados das previsões dos modelos de aprendizado de máquina.

O processo de seleção de características aplicado pelo algoritmo de Floresta aleatória apontou que os atributos “preço” e “vendas” com defasagens em dias se mostraram os mais eficientes para as previsões.

A Floresta aleatória obteve melhores resultados para as métricas calculadas do RMSE, RMSLE, MAE e R^2 nas previsões de preços e demanda. A variação entre os resultados obtidos pelo modelo SVR quando comparado a Floresta aleatória foram bem pequenas para previsão de preços. A Regressão linear apresentou os piores resultados para previsões de preços e demandas. Além disso, os modelos de Redes Neurais Artificiais LSTM e MLP foram os que demandaram maior custo computacional para treinamento.

Desta forma, pode-se inferir que o modelo de Floresta aleatória se mostrou capaz de realizar previsões diárias de preços e demandas com um baixo erro de RMSE, RMSLE e MAE.

5.1 Propostas de Continuidade

Posteriores trabalhos precisam ser realizados para complementar o desempenho das previsões de preços e demandas. Desta forma, sugere-se:

- Agrupar produtos com desempenho de demanda similares utilizando aprendizado não supervisionado;
- Explorar recursos para séries temporais de demandas com períodos intermitentes.
- Explorar recursos externos que possam influenciar na demanda e preço de produtos do varejo.

REFERÊNCIAS

AGGARWAL, C. C. **Neural Networks and Deep Learning: A Textbook**. 1st ed. Cham: Springer, 2018. 497 p.

AHA, D. W.; KIBLER, D.; ALBERT, M. K. Instance-Based Learning Algorithms**. Machine Learning**, Leuven, v. 6, n. 1, p. 37-66, 1991.

ALMEIDA, F. C.; PASSARI, A. F. L. Previsão de vendas no varejo por meio de redes neurais. **Revista de Administração**, São Paulo, v. 43, n. 3, p. 257-272, 2006.

ANDREONI, W.; YIP, S. **Handbook of Materials Modeling. Methods: Theory and Modeling**. 2. ed. Switzerland: Springer, 2020. 1987 p.

AWAN, M. J. *et al*. A Big Data Approach to Black Friday Sales. **Intelligent Automation and Soft Computing**, San Antonio, v. 27, n. 3, p. 785-797, 2021.

BAHADORI, M. T; LIU, Y. Granger Causality Analysis in Irregular Time Series. In: **Proceedings of the 2012 SIAM International Conference on Data Mining**, California, 2012. p. 660-671.

BELGIU, M.; DRAGUT, L. Random Forest in remote sensing: A review of applications and future directions. **ISPRS Journal of Photogrammetry and Remote Sensing**, Amsterdam, v.114, p. 24–31, 2016.

BIAU, G.; SCORNET, E. A random forest guided tour. **TEST**, New York, v. 25, n. 2, p. 197-227, 2016.

BISHOP, C. M. **Neural networks for pattern recognition**. Oxford: Oxford University, 1997. 482 p.

BOUZADA, M. A. C. Aprendendo Decomposição Clássica: Tutorial para um Método de Análise de Séries Temporais. **Tecnologias de Administração e Contabilidade**. Rio de Janeiro, v. 2, n. 1, p.1-18, 2012.

BRAGA, A.; CARVALHO, A.; LUDERMIR, T. **Redes Neurais Artificiais: Teoria e Aplicações**. 2. ed. Rio de Janeiro: Livros Técnicos e Científicos, 2007. 226 p.

BREIMAN, L. Random Forests. **Machine Learning**, Dordrecht, v. 45, p. 5-32, 2001.

BROCKWELL, P. J.; DAVIS, R. A. **Introduction to Time Series and Forecasting**. 2nd ed. New York: Springer-Verlag, 2002. 437 p.

BUSSETI, E.; OSBAND, I.; WONG, S. Deep Learning for Time Series Modeling. **Tech. Rep. CS229**, 2012. Disponível em: <http://cs229.stanford.edu/proj2012/BussetiOsbandWong-DeepLearningForTimeSeriesModeling.pdf>. Acesso em: 07 jun. 2021.

CAO, X. H.; STOJKOVIC, I.; OBRADOVIC, Z. A robust data scaling algorithm to improve classification accuracies in biomedical data. **BMC Bioinformatics**, London, v. 17, n. 359, p. 1-10, 2016.

CARRARA, A. F.; CORREA, A. L. O regime de metas de inflação no Brasil: uma análise empírica do IPCA. **Revista de Economia Contemporânea**, Rio de Janeiro, v.16, n. 3, p. 441-462, 2012.

CASTRO, L. N.; FERRARI, D. G. **Introdução à mineração de dados: conceitos básicos, algoritmos e aplicações**. 1. ed. São Paulo: Saraiva, 2016. 376 p.

CATAL, C.; ECE, K.; ARSLAN, B.; AKBULUT, A. Benchmarking of Regression and Time Series Analysis Techniques for Sales Forecasting. **Balkan Journal of Electrical e Computer Engineering**, Batman, v. 7, n. 1, 2019.

CAUDILL, M. Neural Networks primer, part. I. **AI Expert.** Lawrence**,** v. 2, n. 12, p. 46-52, 1987.

CHAND, Y.; ALAM, M. A.; TEJASWINI, Y. R. S. N. Performance comparison of artificial neural networks learning algorithms and activation functions in predicting severity of autism. **Network Modeling Analysis in Health Informatics and Bioinformatics**, v. 4, n. 1, p. 2, 2015.

CHANDRASHEKAR, G.; SAHIN, F. A survey on feature selection methods. **Computers and Electrical Engineering**. v. 40, n. 1, p. 16-28, 2014.

CHATFIELD, C. The Future of Time-Series Forecasting. **International Journal of Forecasting**. Amsterdam, v. 4, n. 3, p. 411-419, 1988.

CHOLLET, F. **Keras**. 2015. Disponível em: <https://keras.io>. Acesso em: 11 mai. 2019.

CORRAR, L. J.; THEÓPHILO, C. R. **Pesquisa Operacional para decisão em contabilidade e administração**. São Paulo: Atlas, 2004. 490 p.

CORRÊA, H. L.; CORRÊA, C. A. **Administração de produção e operações - manufatura e serviços: uma abordagem estratégica**. 4. ed. São Paulo: Atlas, 2019. 591 p.

CORTES, C.; VAPNIK, V. Support-vector networks. **Machine Learning**, Dordrecht, v. 20, n. 3, p. 273-297, 1995.

CREPALDI, P. G. **Um estudo sobre a árvore de decisão e sua importância na habilidade de aprendizado**. Universidade Católica de Londrina, 2012. Disponível em: <https://www.inesul.edu.br/revista/arquivos/arq-idvol_15_1320100263.pdf>. Acesso em: 13 mai. 2019.

CRIMINISI, A *et al*. Regression forests for efficient anatomy detection and localization in computed tomography scans. **Medical image analysis**, Amsterdam, v. 17, n. 8, p. 1293-1303, 2013.

CUESTA, H.; KUMAR, D. S. **Practical Data Analysis**. 2nd ed. Birmingham: Packt Publishing, 2016. 306 p.

CYBENKO, G. Approximation by Superpositions of a Sigmoidal Function. **Mathematics of Control, Signals, and Systems**, v. 2, p. 303-314, 1989.

DEMSAR, J. *et al*. Orange: Data Mining Toolbox in Python. **Journal of Machine Learning Research**, Brookline, v.14, p. 2349-2353, 2013.

DIETTERICH, T. G. Ensemble methods in machine learning. **Multiple Classifier Systems Lecture Notes in Computer Science**, v. 1857, p. 1-15, 2000.

DIETTERICH, T. G. Machine Learning for Sequential Data: A Review. **Structural, Syntactic, and Statistical Pattern Recognition (SSPR)**, Berlin, v. 2396, p. 15-30, 2002.

DOGANIS, P.; ALEXANDRIDIS, A.; PATRINOS, P.; SARIMVEIS, H. Time Series sales forecasting for short shelf-life food products based on artificial neural networks and evolutionary computing. **Journal of Food Engineering**, Oxford, v. 75, p. 196-204, 2006.

EHLERS, R. S. **Análise de Séries Temporais**. 5. ed. 2009. Disponível em: <http://conteudo.icmc.usp.br/pessoas/ehlers/stemp/stemp.pdf>. Acesso em: 27 abr. 2019.

FACELI, K. *et al*. **Inteligência Artificial: Uma Abordagem de Aprendizado de Máquina**. 1. ed. Rio de Janeiro: Livros Técnicos e Científicos Editora Ltda, 2011. 378 p.

FAUSETT, L. **Fundamentals of Neural Networks: architectures, algorithms, and applications**. New Jersey: Prentice-Hall, 1994. 461 p.

FENG, C. *et al*. Log-transformation, and its implications for data analysis. **Shangai Arch Psychiatry**, v. 26, n. 2, p. 105-109, 2014.

FERNEDA, E. Redes neurais e sua aplicação em sistemas de recuperação de informação. **Ci Inf**, Brasília, v. 35, n. 1, p. 25-30, 2006.

FRIEDL, M. A.; BRODLEY, C. E. Decision tree classification of land cover from remotely sensed data. **Remote Sensing of Environment**, New York, v. 61, n. 3, p. 399-409, 1997.

FRIEDMAN, M. The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. **Journal of the American Statistical Association**, Alexandria, v. 32, n. 200, p. 675-701, 1937.

GENUER, R. *et al*. Random Forests for Big Data. **Big Data Research**. Amsterdam, v. 9, p. 28-46, 2017.

GEURTS, M. D.; KELLY, J. P. Forecasting retail sales using alternative models. **International Journal of Forecasting**, Amsterdam, v. 2, n. 3, p.261-272, 1986.

GORDAZI, A.; AMIRI, R.; TALAEI, A.; JAMASB, T. Predicting oil price movements: A dynamic Artificial Neural Network approach**. Energy Policy**, Oxford, v. 68, p. 371-382, 2014.

GRECO, A.; AREND, L. **Contabilidade: teoria e prática básicas**. 4. Ed. São Paulo: Saraiva, 2013. 552 p.

GREFF, K. *et al*. LSTM: A Search Space Odyssey. **IEEE Transactions on Neural Networks and Learning Systems**, Piscataway, v. 28, n. 10, p. 2222-2232, 2015.

GUO, G. *et al*. KNN Model-Based Approach in Classification. In: **On the Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE. OTM 2003.** Berlin, 2003. P. 986-996.

GUO, H.; YIN, J.; ZHAO, J.; YAO, L.; XIA, X.; LUO, H. An Ensemble Learning for predicting Breakdown Field Strength of Polyimide Nanocomposite Films. **Journal of Nanomaterials**, London, v.15, p. 1-11, 2015.

GUO, T. *et al*. Robust Online Time Series Prediction with Recurrent Neural Networks. In: **IEEE International Conference on Data Science and Advanced Analytics**, 2016, p. 816-825.

GURAJATI, D. N.; PORTER, D. C. **Econometria Básica**. 5. ed. Porto Alegre: AMGH Editora Ltda, 2011. 924 p.

GUYON, I.; ELISSEEFF, A. An Introduction of Variable and Feature Selection. **Journal of Machine Learning Research**, Brookline, v. 3, p. 1157-1182, 2003.

HADAVANDI, E.; SHAVANDI, H.; GHANBARI, A. An improved sales forecasting approach by integration of genetic fuzzy systems and data clustering: Case study of printed circuit board. **Expert Systems with Applications**, Oxford, v. 38, n. 8, p. 9392-9399, 2011.

HAN, J.; KAMBER, M. **Data Mining: Concepts and Techniques**. Massachusetts: Morgan Kaufmann Publishers, 2006. 744 p.

HARRIS, C. R. *et al*. Array programming with NumPy. **Nature**, Berlin, v. 585, p. 357-362, 2020.

HAYKIN, S. **Redes Neurais: Princípios e Prática**. 2. ed. Porto Alegre: Bookman, 2001. 900 p.

HEBB, D. O. **The Organization of Behavior: A Neuropsychological Theory**. New York: J. Wiley e Sons Inc, 1949. 378 p.

HOCHREITER, S.; SCHMIDHUBER, J. Long Short-Term Memory. **Neural Computation**, Cambridge, v. 9, n. 8, p. 1735-1780, 1997.

HOPFIELD, J. Neural Networks, and physical systems with emergent collective computational abilities. **Proceedings of the National Academy of Sciences of the United States of America**, Washington, v. 79, p. 2554-2558, 1982.

HOTA, H. S.; HANDA, R.; SHRIVAS, A. K. Time Series Data Prediction Using Sliding Window Based RBF Neural Network**. International Journal of Computational Intelligence Research**, v.13, n. 5, p. 1145-1156, 2017.

HUYNH, A. N. L. *et al.* Near Real-Time Global Solar Radiation Forecasting at Multiple Time-Step Horizons Using Long Short-Term Memory Network. **Energies**, Basel, v. 13, n. 14, 2020.

INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA (IBGE). **Índice Nacional de Preços ao Consumidor Amplo – IPCA**. Disponível em: <https://www.ibge.gov.br/estatisticas/economicas/precos-e-custos/9256-indice-nacional-de-precos-ao-consumidor-amplo.html?=&t=o-que-e/>. Acesso em: 15 nov. 2019.

INSTITUTO DE PESQUISA ECONÔMICA APLICADA (IPEA). **IPCA – geral – índice (dez. 1993 = 100)**. Disponível em: <http://www.ipeadata.gov.br/Default.aspx>. Acesso em: 20 nov. 2019.

ISHWARAN, H.; LU, M. Standard errors and confidence intervals for variable importance in random forest regression, classification, and survival. **Statistics in Medicine**, Hoboken, v. 38, n. 4, p. 1-25, 2018.

IUDÍCIBUS, S.; MARION, J. C. **Contabilidade comercial: atualizado conforme Lei nº 11.638/07 e Lei nº 11.941/09**. 10 ed. São Paulo: Atlas, 2016. 463 p.

JABBAR, H. K.; KHAN, R. Z. Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study). **Computer Science, Communication, and Instrumentation Devices**, 2014. Disponível em: <https://www.researchgate.net/profile/Haider_Allamy/publication/295198699_METHODS_TO_AVOID_OVER-FITTING_AND_UNDER-FITTING_IN_SUPERVISED_MACHINE_LEARNING_COMPARATIVE_STUDY/links/56c8253f08aee3cee53a3707.pdf >. Acesso em: 21 abr. 2020.

JIANGUANG, D.; JIRUTITIJAROEN, P. Short-Term Load Forecasting Using Time Series Analysis: A Case Study for Singapore. In: **IEEE Conference on Cybernetics and Intelligent Systems**, Singapura, 2010, p. 231-236.

JOHH, G. H.; KOHAVI, R.; PLEGER, L. Irrelevant Features and the Subset Selection Problem. In: **Proceedings of the Eleventh International Conference on Machine Learning,** New Brunswick, 1994, p. 121-129.

JOSHI, S.; DIETTERICH, T. G. **Calibrating recurrent sliding window classifiers for sequential Supervised learning**. 2003. Disponível em:<https://ir.library.oregonstate.edu/concern/parent/vt150k44w/file_sets/7h149r02k>. Acesso em: 13 fev. 2020.

KINGMA, D. P; BA, J. L. **Adam: A method for Stochastic Optimization**. 2015. Disponível em:<https://arxiv.org/pdf/1412.6980.pdf>. Acesso em: 01 jun. 2021.

KOTLER, P.; KELLER, K. L. **Administração de Marketing**. 12. ed. São Paulo: Prentice Hall, 2006. 750 p.

KRAJEWSKI, L.; RITZMAN, L.; MALHOTRA, M. **Administração de produção e operações**. 8. ed. São Paulo: Pearson Prentice Hall, 2009. 636 p.

KUHN, M.; JOHNSON, K. **Feature Engineering and Selection: A Practical Approach for Predictive Models**. 1st. ed. Florida: Chapman and Hall/CRC, 2019. 310 p.

LAPPONI, J. C. **Estatística usando Excel**. Rio de Janeiro: Elsevier, 2005. 476 p.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep Learning. **Nature International journal of Science**, v. 521, p. 436-444, 2015.

LUGER, G. F. **Inteligência Artificial**. 6. ed. São Paulo: Pearson Education do Brasil, 2013. 632 p.

LUO, C. *et al*. Prediction of Vegetable Price Based on Neural Network and Genetic Algorithm. **Computer and Computing Technologies in Agriculture IV. CCTA**, v. 386, p. 672-681, 2010.

MARIANO, J. **Introdução à economia brasileira**. 2. ed. São Paulo: Saraiva, 2012. 112 p.

MARION, J. C. **Contabilidade básica**. 11. ed. São Paulo: Atlas, 2015. 280 p.

MARTINEZ, T. S.; CERQUEIRA, V. S. Estrutura da inflação brasileira: Determinantes e desagregação do IPCA. **Economia e Sociedade**, Campinas, v. 22, n. 2, p. 409-456, 2013.

MAS, J. F.; FLORES, J. J. The application of artificial neural networks to the analysis of remotely sensed data. **International Journal of Remote Sensing**. v. 29, n. 3, p.617-663, 2008.

MATTAR, F. N. **Administração do varejo**. 1. ed. Rio de Janeiro: Elsevier, 2011. 648 p.

MCCULLOCH, W.S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **Bulletin of Mathematical Biophysics**, New York, v. 5, p. 115-133, 1943.

MCKINNEY, W. **pandas: a Foundational Python Library for Data Analysis and Statistics**. [S.l.: s.n], 2011. Disponível em: <https://dlr.de/sc/portaldata/15/resources/dokumente/pyhpc2011/submissions/pyhpc2011_submission_9.pdf>. Acesso em: 07 jun. 2021.

MELO, F. A. M. IGP ou IPCA: como estimadores da inflação e indexadores na economia brasileira. **Revista de Economia Política**, São Paulo, v. 6, n. 2, p. 100-108, 1986.

MINSKY, M. L.; PAPERT, S.A. **Perceptrons**. Cambridge: MIT, 1960.

MONTGOMERY, D. C.; PECK, E. A.; VINING, G. G. **Introduction to Linear Regression Analysis**. 5th ed. [S.l.]: Wiley, 2012. 672 p.

MONTGOMERY, D. C.; RUNGER, G. C. **Applied Statistics and Probability for Engineers**. 6th ed. [S.l.]: John Wiley & Sons, 2013. 811 p.

MOON, M. A.; MENTZER, C.; SMITH, C. D. Conducting a sales forecasting audit. **International Journal of Forecasting**, Amsterdam, v. 19, p. 5-25, 2003.

MORAIS, Nyedja Fialho. **Análise de regressão linear com estudo de caso em acidentes de trânsito**. 2010. Monografia (Bacharel em Estatística), Universidade Estadual da Paraíba, Campina Grande, Paraíba, 2010.

MORETTIN, P. A.; TOLOI, C. M. C. **Análise de séries temporais**. 2. ed. São Paulo: Editora Edgard Blucher Ltda, 2006. 564 p.

MUKAKA, M. M. A guide to appropriate use of Correlation coefficient in medical research. **Malawai medical journal: the journal of Medical Association of Malawi**, v. 24, n. 3, p. 67-71, 2012.

NEMENYI, P. **Distribution-free Multiple Comparisons**. Princeton: Princeton University, 1963. 254 p.

NIKOLOPOULOS, K.; PETROPOULOS, F. Forecasting for big data: Does suboptimality matter? **Computers and Operations Research**, [S.l.], v. 98, p. 322-329, 2018.

NOGAMI, O.; PASSOS, C. R. M. **Princípios de economia**. 7. ed. São Paulo: Cengage Learning, 2016. 670 p.

OSINGA, D. **Deep Learning Cookbook**. 1st ed. [S.l.]: O'Reilly, 2018, 252 p.

PACHECO, A. K vizinhos mais próximos – KNN. **Computação inteligente**, 2017. Disponível em: <http://computacaointeligente.com.br/algoritmos/k-vizinhos-mais-proximos/>. Acesso em: 01 nov. 2019.

PADOVEZE, C. L. **Contabilidade de custos, teoria, prática, Integração com Sistemas de Informações (ERP)**. 3. ed. São Paulo: Cengage Learning, 2013. 510 p.

PAL, A.; PRAKASH, P. **Practical Time Series Analysis: Master Time Series Data Processing, Visualization, and Modeling using Python**. Birmingham: Packt Publishing, 2017. 330 p.

PARENTE, J.; BARKI, E. **Varejo no Brasil: Gestão e Estratégia**. 2. ed. São Paulo: Atlas, 2014. 423 p.

PEDREGOSA, F. *et al*. Scikit-learn: Machine Learning in Python. **Journal of Machine Learning Research**, Brookline, v. 12, p. 2825-2830, 2011.

PETNEHÁZI, G. **Recurrent Neural Networks for Time Series Forecasting**, 2019. Disponível em: <https://arxiv.org/pdf/1901.00069.pdf>. Acesso em: 18 de mar. 2021.

RAMOS, P.; SANTOS, N.; REBELO, R. Performance of state space and ARIMA models for consumer retail sales forecasting. **Robotics and Computer-Integrated Manufacturing**, Oxford, v.34, p.151-163, 2015.

NAU, R. **Inflation adjustment**. Durham, 2020. Disponível em: <http://people.duke.edu/~rnau/411infla.htm>. Acesso em: 19 de mar. 2021.

ROSENBLATT, F. The Perceptron: a probabilistic model for information storage and organization in the brain. **Psychological Review**, Washington, v. 65, p. 386-408, 1958.

ROSIN, P. L.; FIERENS, F. Improving Neural Network Generalization. In: **International Geoscience and Remote Sensing Symposium**.1995, p. 1255-1257.

ROSSUM, G. V.; DRAKE, F. L. **The Python Language Reference Manual**. [S.l.]: Network Theory ltda, 2011, 150 p.

RUMELHART, D. E.; HINTON, G. E; WILLIANS, R. J. Learning representations by back-propagating errors. **Nature Publishing Group**, v. 323, p. 533-536, 1986.

RUSSEL, S.; NORVIG, P. **Inteligência Artificial**. 3. ed. Rio de Janeiro: Elsevier, 2013.

SAEYS, Y.; INZA, I.; LARRAÑAGA, P. A review of feature selection techniques in bioinformatics. **Bioinformatics**, Oxford, v. 23, n. 19, p. 2507-2517, 2007.

SALAKEN, S. M.; HOSEN, M. A.; KHOSRAV, A.; NAHAVANDI, S. Forecasting Bike Sharing Demand Using Fuzzy Inference Mechanism. **Neural Information Processing**, p. 567-574, 2015.

SALHI, D. E.; TARI, A.; KECHADI, MT. Using Machine Learning for Heart Disease Prediction. In: **International Conference on Computing Systems and Applications**. Cham, 2021, p. 70-81.

SANTOS, A. M. M. S.; COSTA, C. S. Características gerais do varejo no Brasil. Brasília: BNDES. **Produção BNDES**, 1997. Disponível em <https://web.bndes.gov.br/bib/jspui/bitstream/1408/7125/2/BS%2005%20Caracteristicas%20gerais%20do%20varejo%20no%20Brasil_P.pdf>. Acesso em: 01 de dez. 2019.

SANTOS, A. M.; SEIXAS, J.M.; PEREIRA, B. B.; MEDRONHO, R. A. Usando Redes Neurais Artificias e Regressão Logística na predição da Hepatite A. **Revista Brasileira de Epidemiologia**, São Paulo, v.8n.2, p. 117-126, 2005.

SANTOS, A. S. **Contabilidade**. 1. ed. São Paulo: Pearson Education do Brasil, 2014.

SAPANKEVYCH, N.; SANKAR, R. Time Series Prediction Using Support Vector Machines: A Survey. **IEEE Computational Intelligence Magazine**, Piscataway, v. 4, n. 2, p. 24-38, 2009.

SHOTTON, J. *et al*. Real-time human pose recognition in parts from single depth images. **Communications of the ACM**, New York, v. 56, n. 1, p. 116–124, 2013.

SILVA, D. F.; SILVA, R. A. **Fundamentos da economia**. Porto Alegre: SAGAH, 2018. *E*-Book. ISBN 978-85-9502-833-3.

SOCIEDADE BRASILEIRA DE VAREJO E CONSUMO. **Estudo revela importante papel do Varejo na Economia Brasileira**. Brasília. 2018. Disponível em: <http://sbvc.com.br/estudo-papel-varejo-2018/>. Acesso em: 20 Mar. 2019.

SONG, X.; HUANG, J.; SONG, D. Air Quality Prediction based on LSTM-Kalman Model. **IEEE Joint International Information Technology and Artificial Intelligence Conference**, 2019, p. 695-699.

SPILIOTIS, E. *et al*. Comparison of statistical and machine learning methods for daily SKU demand forecasting. **Operational Research**, Heidelberg, 2020.

STÁNCZYK, U.; JAIN, L. C. **Feature selection for data and Pattern Recognition**. New York: Springer, 2015. 355 p.

SUYKENS, J. A. K.; VANDEWALLE, J. Least Squares Support Vector Machine Classifiers. **Neural Processing Letters**, Dordrecht, v. 9, n. 3, p. 293-300, 1999.

TSOUMAKAS, G. A survey of machine learning techniques for food sales prediction. **Artificial Intelligence Review**, v.52, p. 441-447, 2019.

TUBINO, D. F. **Planejamento e controle da produção: teoria e prática**. 2. ed. São Paulo: Atlas, 2007. 190 p.

VAROTTO, L. F. História do Varejo. **GV-executivo**, São Paulo, v. 5, n. 1, p. 86-90. 2006.

VASCONCELOS, M. A. S.; GARCIA, M. E. **Fundamentos de Economia.** 6. ed. São Paulo: Saraiva, 2019. ISBN 978-85-53131-74-7.

VIRTANEN, P. *et al*. fundamental algorithms for scientific computing in Python. **Nat Methods**, v. 17, p. 261-272, 2020.

WEIGEND, A. S.; GERSHENFELD, N. A**. Time Series Prediction: Forecasting the Future and Understanding the Past**. 1st. ed. Colorado: Westview Press, 1993. 672 p.

WETTSCHERECK, D.; DIETTERICH, T. G. Locally Adaptive Nearest Neighbor Algorithms. **Neural Information Processing Systems**, v. 6, p. 184-191, 1994.

WIDROW, B.; HOFF, M. E. Adaptative witching circuits. In: **IRE WESCON Convention Record**, Los Angeles, p. 96-104, 1960.

WOLFE, F.; MICHAUD, K. Predicting Depression in Rheumatoid Arthritis: The Signal Importance of Pain Extent and Fatigue, and Comorbidity. **Arthritis & Rheumatoid**, v. 61, n. 5, p. 667-673, 2009.

YAFFE, R. A.; MCGEE M. **Introduction to Time Series Analysis and Forecasting with applications of SAS and SPSS**. San Diego: Academic Press, 2002. 555 p.

YAN, X.; SU, X. G. **Linear regression analysis: theory and computing**. [S.l.]: World Scientific Publishing Company, 2009. 348 p.

YANG, R. **Omphalos, Uber's Parallel and Language-Extensible Time Series Backtesting tool.** 2018. Disponível em: <https://eng.uber.com/omphalos/>. Acesso em: 04 Mai. 2021.

YUE, L.; YAFENG, Y.; JUNJUN, G.; CHONGLI, T. Demand Forecasting by Using Support Vector Machine. In: **Third International Conference on Natural Computation (ICNC 2007)**, China. 2007. p. 24-27.

APÊNDICE A – Código Fonte

```
import pandas                       as pd
import numpy                        as np
from sklearn.preprocessing          import MinMaxScaler
from sklearn.metrics                import (mean_absolute_error, mean_squared_error,
                                            r2_score, mean_squared_log_error)
from sklearn.ensemble               import RandomForestRegressor
from sklearn.neighbors              import KNeighborsRegressor
from sklearn.linear_model           import LinearRegression
from sklearn.svm                    import SVR
from keras.models                   import Sequential
from keras.layers                   import Dense, Dropout, LSTM
import os
import datetime
from dateutil.relativedelta         import *
from tqdm                           import tqdm_notebook
from datetime import datetime
import time


def ExpandingWindowTs(df,past_step=546,future_step=90,field_data='DATA'):
    train indices list = []
    test_indices_list  = []
    validation_exp_date = None

    if (validation_exp_date == None):
        validation_exp_date = df[field_data].min().date() + \
                              relativedelta(days=past_step)

    start_train = df[field_data].min().date()
    end_train = validation_exp_date

    start_test = validation_exp_date + relativedelta(days=1)
    end_test = start_test + relativedelta(days=future_step)
    train_indices_list =[]
    test_indices_list =[]

    while end_test <= df[field_data].max().date():
        indice_train = list(df[(df[field_data].dt.date>=start_train) &
                               (df[field_data].dt.date<=end_train)].index)
        indice_test = list(df[ (df[field_data].dt.date>=start_test) &
                                (df[field_data].dt.date<=end_test)].index)

        train_indices_list.append(indice_train)
        test_indices_list.append(indice_test)
```

```
        start_train = start_train
        end_train =  end_train + relativedelta(days=future_step+1)
        start_test = end_train + relativedelta(days=1)
        end_test = start_test + relativedelta(days=future_step)

    index output = [(train,test) for train,test in zip(train indices list,test indices list)]

    return index_output

def add_inflacao(ipd,row,field):
  ipeadata = ipd.copy()
  ipeadata['Indice'].astype('object')
  indice_passado = ipeadata[ipeadata['Data'] == row['EMISSAO_INDICE']]
  indice_atual  = ipeadata[ipeadata['Data'] == '2019.12']
  corrigido = indice_atual.iloc[:,1].values/indice_passado.iloc[:,1].values

  return np.round((row[field] * corrigido),2).item()

def rmsle(y_true,y_pred):
  return np.sqrt(mean_squared_log_error(y_true, y_pred))

def create_sales_lag(df, gpby_cols, target_col, lags):
  if not gpby_cols:
    gpby = df.copy()
  else:
    gpby = df.groupby(gpby_cols)
  for i in lags:
    df[target_col+'_lag_{}'.format(i)] = \
    gpby.groupby('id_produto')[target_col].shift(i)

  return df

def feature_engineering_sazonality(df):
  dfs = df.copy()
  dfs['trimestre'] = dfs['data'].dt.quarter
  dfs['week_year'] = dfs['data'].dt.isocalendar().week
  dfs['week_day']  = dfs['data'].dt.weekday
  dfs['final de semana'] = dfs['data'].dt.weekday//5
  dfs['final de semana'] = dfs['final de semana'].astype(int)
  dfs['mes'] = dfs['data'].dt.month
  dfs['dia'] = dfs['data'].dt.day
  dfs['Ano'] = dfs['data'].dt.year

  return dfs

def transform_cyclic_variables(df):
  dft = df.copy()

  dft['preco'] = np.log1p(dft['preco'])
```

```
  dft['custo'] = np.log1p(dft['custo'])

  dft['mes_sin'] = dft['mes'].apply(lambda x: np.sin(x * (2*np.pi/12)))
  dft['mes_cos'] = dft['mes'].apply(lambda x: np.cos(x * (2*np.pi/12)))

  dft['dia sen'] = dft['dia'].apply(lambda x: np.sin(x * (2*np.pi/30)))
  dft['dia_cos'] = dft['dia'].apply(lambda x: np.cos(x * (2*np.pi/30)))

  dft['dia_semana_sen'] = dft['week_day'].apply(lambda x: np.sin(x * (2*np.pi/7)))
  dft['dia_semana_cos'] = dft['week_day'].apply(lambda x: np.cos(x * (2*np.pi/7)))

  dft['semana_ano_sen'] = dft['week_year'].apply(lambda x: np.sin(x * (2*np.pi/52)))
  dft['semana_ano_cos'] = dft['week_year'].apply(lambda x: np.cos(x * (2*np.pi/52)))

  dft['trimestre_sen'] = dft['trimestre'].apply(lambda x: np.sin(x * (2*np.pi/4)))
  dft['trimestre_cos'] = dft['trimestre'].apply(lambda x: np.cos(x *(2*np.pi/4)))

  return dft

def datapreparation(df,field,transform=True):
  trains = df.copy()
  trains = create_sales_lag(trains,None,field,lags=[i for i in range(1,31)])
  trains = feature_engineering_sazonality(trains)
  if transform:
    trains = transform_cyclic_variables(trains)
    trains.drop(['mes','dia','trimestre',
                 'week_year','week_day'],axis=1,inplace=True)

  return trains

def add_inflacao(ipd,row,field):
  ipeadata = ipd.copy()
  ipeadata['Indice'].astype('object')
  indice_passado = ipeadata[ipeadata['Data'] == row['EMISSAO_INDICE']]
  indice_atual  = ipeadata[ipeadata['Data'] == '2019.12']
  corrigido = indice_atual.iloc[:,1].values/indice_passado.iloc[:,1].values

  return np.round((row[field] * corrigido),2).item()

def load_ipeadata():
  ipeadata = pd.read_csv(path+'/ipeadata[09-02-2021-07-02].csv',header=None)
  ipeadata.drop(0,axis=0,inplace=True)
  ipeadata.drop(2,axis=1,inplace=True)
  ipeadata.columns = ['Data','Indice']
  ipeadata.dtypes
  ipeadata['Indice'] = ipeadata['Indice'].astype(float)

def feature importance random forest(df,field,plot graphic=False):
  rf = RandomForestRegressor(n_estimators=1000)
```

```
  rf.fit(df.drop([field],axis=1),df[field].values.ravel())

  lista_atributos    = list(df.drop([field],axis=1).columns)
  lista_import = list(rf.feature_importances_)
  importancia_atributo = [(feature, importance) for \
                          feature, importance in zip(lista atributos, lista import)]
  importancia_atributo = sorted(importancia_atributo, key = lambda x: x[1], reverse = True)
  if plot_graphic:
    plt.figure(figsize=(15,3))
    x_val = list(range(len(lista_import)))
    importancias_classificadas=[importance[1]*100 for importance in importancia_atributo]
    atributos_class = [importance[0] for importance in importancia_atributo]
    cumulative_importances = np.cumsum(importancias_classificadas)
    plt.plot(x_val, cumulative_importances, color='b',marker='o')
    plt.hlines(y = (0.95)*100, xmin=0, \
               xmax=len(importancias_classificadas),color='r',linestyles='dashed')
    plt.xticks(x_val, atributos_class,rotation=90)
    plt.xlabel('Atributo',weight='bold');
    plt.ylabel('% Importância cumulativa',weight='bold');
    plt.grid(True)
    plt.show()

def rmsle(y_true,y_pred):
  return np.sqrt(mean_squared_log_error(y_true, y_pred))

def load_data(price_demand):
  if price_demand=='price':
    return pd.read_parquet('price_20full.parquet')
  else:
    return pd.read_parquet('vendas_20full.parquet')

def apply_models_price():
  df2= load_data('price')
  df2['preco'] = df2['preco_inflacionado']
  df2['custo'] = df2['custo_inflacionado']
  df2 = df2.drop(['preco_inflacionado','custo_inflacionado'],axis=1)

  indexs=ExpandingWindowTs(df2,past_step=546,future_step=90,field_data='data')
  fe_selecionadas = ['preco_lag_1','preco_lag_2','preco_lag_3','preco_lag_4',
                     'preco_lag_5','custo','id_produto','data','preco','descricao']
  models = [('SVR',SVR(C=1)),
            ('KNN',KNeighborsRegressor(n_neighbors=17)),
            ('RF', RandomForestRegressor(n_estimators=1000,min_samples_split=2)),
            ('Reg_Linear',LinearRegression()),
            ('MLP_KERAS',None),
            ('LSTM',None)]
  dfs = pd.DataFrame()
  df rank = pd.DataFrame()
  df_metrics =pd.DataFrame()
```

```
for train_indice, test_indice in tqdm_notebook(indexs,total=len(indexs),position=0):
  for index,mdl in enumerate(models):

    preco_lag_1_mms = MinMaxScaler()
    preco lag 2 mms = MinMaxScaler()
    preco_lag_3_mms = MinMaxScaler()
    preco_lag_4_mms = MinMaxScaler()
    preco_lag_5_mms = MinMaxScaler()
    price_scaler = MinMaxScaler()

    xtreino = datapreparation(df2.loc[train_indice],'preco')
    xtreino = xtreino[fe_selecionadas].dropna()

    DT_INI_TRAIN, DT_FIN_TRAIN = \
      xtreino.data.min().strftime('%y-%m-%d'),xtreino.data.max().strftime('%y-%m-%d')
    ytreino = xtreino['preco'].values
    xtreino.drop('preco',axis=1,inplace=True)
    xtreino.drop(['id_produto','descricao','data'],axis=1,inplace=True)
    y_train = price_scaler.fit_transform( np.log1p(ytreino).reshape(-1,1)).ravel()

    xteste = datapreparation(df2.loc[test_indice],'preco')
    xteste = xteste[fe_selecionadas].dropna()
    bkp = xteste.copy()
    DT_INI_TESTE, DT_FIN_TESTE = \
      xteste.data.min().strftime('%y-%m-%d'),xteste.data.max().strftime('%y-%m-%d')

    y_test = xteste['preco'].values
    xteste.drop('preco',axis=1,inplace=True)
    xteste.drop(['id_produto','descricao','data'],axis=1,inplace=True)

    xtreino['preco_lag_1'] = preco_lag_1_mms.fit_transform(xtreino[['preco_lag_1']])
    xtreino['preco_lag_2'] = preco_lag_2_mms.fit_transform(xtreino[['preco_lag_2']])
    xtreino['preco_lag_3'] = preco_lag_3_mms.fit_transform(xtreino[['preco_lag_3']])
    xtreino['preco_lag_4'] = preco_lag_4_mms.fit_transform(xtreino[['preco_lag_4']])
    xtreino['preco_lag_5'] = preco_lag_5_mms.fit_transform(xtreino[['preco_lag_5']])

    xteste['preco_lag_1'] = preco_lag_1_mms.transform(xteste[['preco_lag_1']])
    xteste['preco_lag_2'] = preco_lag_2_mms.transform(xteste[['preco_lag_2']])
    xteste['preco_lag_3'] = preco_lag_3_mms.transform(xteste[['preco_lag_3']])
    xteste['preco_lag_4'] = preco_lag_4_mms.transform(xteste[['preco_lag_4']])
    xteste['preco_lag_5'] = preco_lag_5_mms.transform(xteste[['preco_lag_5']])

    dt_ini_train = df2.loc[train_indice]['data'].dt.strftime('%Y-%m-%d').min()
    dt_fin_train = df2.loc[train_indice]['data'].dt.strftime('%Y-%m-%d').max()
    dt_ini_valid = df2.loc[test_indice]['data'].dt.strftime('%Y-%m-%d').min()
    dt_fin_valid = df2.loc[test_indice]['data'].dt.strftime('%Y-%m-%d').max()

    print(f'{DT_INI_TRAIN}--{DT_FIN_TRAIN}|{DT_INI_TESTE} \
```

```
        {DT_FIN_TESTE}->Modelo: {mdl[0]}')

start_time = datetime.now()
if mdl[0] == 'LSTM':
    xtreino = xtreino.values
    xtreino = xtreino.reshape((xtreino.shape[0],1,xtreino.shape[1]))
    xteste = xteste.values
    xteste = xteste.reshape((xteste.shape[0],1,xteste.shape[1]))
    model = Sequential()
    model.add(LSTM(units=50,return_sequences=True,activation='relu',
                   input_shape=(xtreino.shape[1],xtreino.shape[2])))
    model.add(Dropout(0.2))
    model.add(LSTM(units=40, return_sequences=True,activation='relu'))
    model.add(Dropout(0.2))
    model.add(LSTM(units=40, return_sequences=False))
    model.add(Dropout(0.2))
    model.add(Dense(units=1,activation='linear'))
    model.compile(optimizer='adam',loss='mse',metrics=['mean_absolute_error'])
    model.fit(xtreino,y_train,epochs=500,batch_size=26,verbose=False)

    y_pred = np.expm1(price_scaler.inverse_transform( model.predict(xteste)))
else:
    if mdl[0] =='MLP_KERAS':
      mlp = Sequential()
      mlp.add(Dense(units=50, input_dim=xtreino.shape[1]))
      mlp.add(Dropout(0.15))
      mlp.add(Dense(50,activation='relu'))
      mlp.add(Dropout(0.15))
      mlp.add(Dense(1,activation='linear'))
      mlp.compile(optimizer='adam',loss='mse', metrics=['mae'])
      mlp.fit(xtreino,y_train, epochs=500,batch_size=16, verbose=False)
      y_pred = np.expm1(price_scaler.
                        inverse_transform(mlp.predict(xteste).reshape(-1,1)))
    else:
      mdl[1].fit(xtreino,y_train)
      y_pred=np.expm1(price_scaler.
                      inverse_transform(mdl[1].predict(xteste).reshape(-1,1)))
dif = datetime.now()-start_time

bkp['y_pred']=y_pred
bkp['modelo']=mdl[0]
df_rank = pd.concat([df_rank,bkp])

error_mae = mean_absolute_error(y_test,y_pred)
rmse=mean_squared_error(y_test,y_pred,squared=True)
m_rmsle = rmsle(y_test,y_pred)
r2 = r2_score(y_test,y_pred)

data= {'Modelo': mdl[0],
```

```
                'RMSE': rmse,
                'MAE'  : error_mae,
                'RMSLE': m_rmsle,
                'r2': r2,
                'periodo_treinamento':
                    df2.loc[train_indice]['data'].dt.strftime('%Y-%m-%d').min()+ \
                    ' a '+ df2.loc[train_indice]['data'].dt.strftime('%Y-%m-%d').max(),
                'periodo_teste':
                    df2.loc[test_indice]['data'].dt.strftime('%Y-%m-%d').min()+ \
                    ' a '+ df2.loc[test_indice]['data'].dt.strftime('%Y-%m-%d').max(),
                'DT_TRAIN_BE': DT_INI_TRAIN +'|'+DT_FIN_TRAIN,
                'DT_TESTE_BE': DT_INI_TESTE +'|'+DT_FIN_TESTE,
                'tempo_execucao': dif}

        df_metrics = df_metrics.append(data,ignore_index=True)
    dfs[mdl[0]] = df_metrics.mean()
    df_metrics.to_csv('price_results.csv')


def apply_models_demand():
    df2 = load_data('demand')
    fe_selecionadas = ['vendas_lag_1','vendas_lag_7','vendas_lag_28',
                       'vendas_lag_21','vendas_lag_2','vendas_lag_14',
                       'dia_semana_sen','preco','vendas_lag_3',
                       'vendas_lag_4','vendas_lag_6','vendas_lag_8',
                       'vendas_lag_13','vendas_lag_20','vendas_lag_27',
                       'dia_semana_cos','id_produto','data','vendas','descricao']
    indexs=ExpandingWindowTs(df2,past_step=546,future_step=90,field_data='data')
    models = [('SVR',SVR(C=2)),
              ('KNN',KNeighborsRegressor(n_neighbors=5)),
              ('RF', RandomForestRegressor(n_estimators=1000,min_samples_split=2)),
              ('MLP_KERAS',None),
              ('Reg_Linear',LinearRegression())]
    dfs = pd.DataFrame()
    df_rank = pd.DataFrame()
    df_metrics =pd.DataFrame()

    for train_indice, test_indice in tqdm_notebook(indexs,total=len(indexs),position=0):
      for index,mdl in enumerate(models):

        vendas_lag_1_mms = MinMaxScaler()
        vendas_lag_2_mms = MinMaxScaler()
        vendas_lag_3_mms = MinMaxScaler()
        vendas_lag_4_mms = MinMaxScaler()
        vendas_lag_6_mms = MinMaxScaler()
        vendas_lag_7_mms = MinMaxScaler()
        vendas_lag_8_mms = MinMaxScaler()
        vendas_lag_13_mms = MinMaxScaler()
        vendas_lag_14_mms = MinMaxScaler()
```

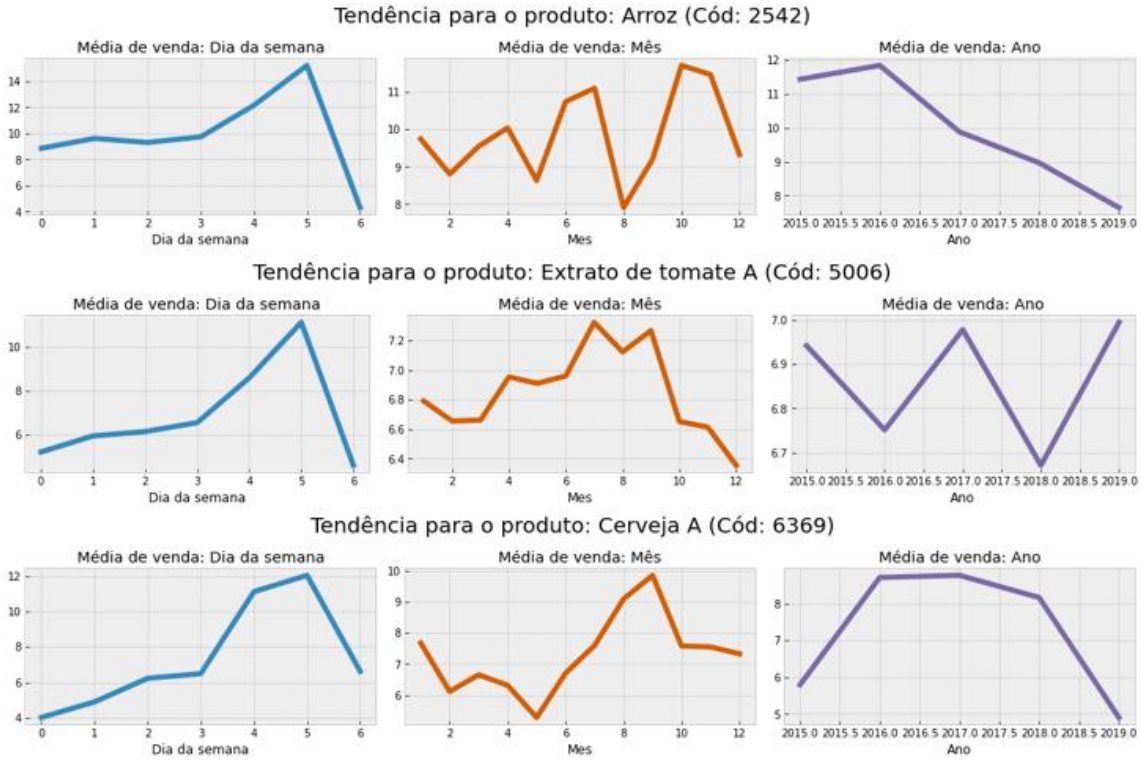
```
vendas_lag_20_mms = MinMaxScaler()
vendas_lag_21_mms = MinMaxScaler()
vendas_lag_27_mms = MinMaxScaler()
vendas_lag_28_mms = MinMaxScaler()
enc_preco_mms     = MinMaxScaler()

sales_scaler = MinMaxScaler(feature_range=(0,1))
xtreino = datapreparation(df2.loc[train_indice],'vendas')
xtreino = xtreino[fe_selecionadas].dropna()

DT_INI_TRAIN, DT_FIN_TRAIN = \
  xtreino.data.min().strftime('%y-%m-%d'),xtreino.data.max().strftime('%y-%m-%d')
ytreino = xtreino['vendas'].values
xtreino.drop('vendas',axis=1,inplace=True)
xtreino.drop(['id_produto','descricao','data'],axis=1,inplace=True)
y_train = sales_scaler.fit_transform(np.log1p(ytreino).reshape(-1,1) ).ravel()

xteste = datapreparation(df2.loc[test_indice],'vendas')
xteste = xteste[fe_selecionadas].dropna()
bkp = xteste.copy()
DT_INI_TESTE, DT_FIN_TESTE = \
  xteste.data.min().strftime('%y-%m-%d'),xteste.data.max().strftime('%y-%m-%d')

y_test = xteste['vendas'].values
xteste.drop('vendas',axis=1,inplace=True)
xteste.drop(['id_produto','descricao','data'],axis=1,inplace=True)

xtreino['vendas_lag_1'] = vendas_lag_1_mms.fit_transform(xtreino[['vendas_lag_1']])
xtreino['vendas_lag_2'] = vendas_lag_2_mms.fit_transform(xtreino[['vendas_lag_2']])
xtreino['vendas_lag_3'] = vendas_lag_3_mms.fit_transform(xtreino[['vendas_lag_3']])
xtreino['vendas_lag_4'] = vendas_lag_4_mms.fit_transform(xtreino[['vendas_lag_4']])
xtreino['vendas_lag_6'] = vendas_lag_6_mms.fit_transform(xtreino[['vendas_lag_6']])
xtreino['vendas_lag_7'] = vendas_lag_7_mms.fit_transform(xtreino[['vendas_lag_7']])
xtreino['vendas_lag_8'] = vendas_lag_8_mms.fit_transform(xtreino[['vendas_lag_8']])
xtreino['vendas_lag_13'] = vendas_lag_13_mms.fit_transform(xtreino[['vendas_lag_13']])

xtreino['vendas_lag_14'] = vendas_lag_14_mms.fit_transform(xtreino[['vendas_lag_14']])
xtreino['vendas_lag_20'] = vendas_lag_20_mms.fit_transform(xtreino[['vendas_lag_20']])

xtreino['vendas_lag_21'] = vendas_lag_21_mms.fit_transform(xtreino[['vendas_lag_21']])
xtreino['vendas_lag_27'] = vendas_lag_27_mms.fit_transform(xtreino[['vendas_lag_27']])
xtreino['vendas_lag_28'] = vendas_lag_28_mms.fit_transform(xtreino[['vendas_lag_28']])

xtreino['preco'] = enc_preco_mms.fit_transform(xtreino[['preco']])

xteste['vendas_lag_1'] = vendas_lag_1_mms.transform(xteste[['vendas_lag_1']])
xteste['vendas_lag_2'] = vendas_lag_2_mms.transform(xteste[['vendas_lag_2']])
xteste['vendas lag 3'] = vendas lag 3 mms.transform(xteste[['vendas lag 3']])
xteste['vendas_lag_4'] = vendas_lag_4_mms.transform(xteste[['vendas_lag_4']])
```

```
xteste['vendas_lag_6'] = vendas_lag_6_mms.transform(xteste[['vendas_lag_6']])
xteste['vendas_lag_7'] = vendas_lag_7_mms.transform(xteste[['vendas_lag_7']])
xteste['vendas_lag_8'] = vendas_lag_8_mms.transform(xteste[['vendas_lag_8']])
xteste['vendas_lag_13'] = vendas_lag_13_mms.transform(xteste[['vendas_lag_13']])
xteste['vendas_lag_14'] = vendas_lag_14_mms.transform(xteste[['vendas_lag_14']])
xteste['vendas lag 20'] = vendas lag 20 mms.transform(xteste[['vendas lag 20']])
xteste['vendas_lag_21'] = vendas_lag_21_mms.transform(xteste[['vendas_lag_21']])
xteste['vendas_lag_27'] = vendas_lag_27_mms.transform(xteste[['vendas_lag_27']])
xteste['vendas_lag_28'] = vendas_lag_28_mms.transform(xteste[['vendas_lag_28']])
xteste['preco'] = enc_preco_mms.transform(xteste[['preco']])

dt_ini_train = df2.loc[train_indice]['data'].dt.strftime('%Y-%m-%d').min()
dt_fin_train = df2.loc[train_indice]['data'].dt.strftime('%Y-%m-%d').max()
dt_ini_valid = df2.loc[test_indice]['data'].dt.strftime('%Y-%m-%d').min()
dt_fin_valid = df2.loc[test_indice]['data'].dt.strftime('%Y-%m-%d').max()

print(f'{DT_INI_TRAIN}--{DT_FIN_TRAIN}|{DT_INI_TESTE} \
       {DT_FIN_TESTE}->Modelo: {mdl[0]}')

start_time = time.time()
if mdl[0] == 'LSTM':
    xtreino = xtreino.values
    xtreino = xtreino.reshape((xtreino.shape[0],1,xtreino.shape[1]))
    xteste = xteste.values
    xteste = xteste.reshape((xteste.shape[0],1,xteste.shape[1]))

    model = Sequential()
    model.add(LSTM(units=50,return_sequences=True,activation='relu',
                   input_shape=(xtreino.shape[1],xtreino.shape[2])))
    model.add(Dropout(0.20))
    model.add(LSTM(units=40, return_sequences=True,activation='relu'))
    model.add(Dropout(0.20))
    model.add(LSTM(units=40, return_sequences=False))
    model.add(Dropout(0.20))
    model.add(Dense(units=1,activation='linear'))
    model.compile(optimizer='adam',loss='mse',metrics=['mean_absolute_error'])
    model.fit(xtreino,y_train,epochs=500,batch_size=26,verbose=False)

    y_pred = np.expm1(sales_scaler.
                      inverse_transform(model.predict(xteste).reshape(-1,1)))
else:
    if mdl[0] =='MLP_KERAS':
      mlp = Sequential()
      mlp.add(Dense(units=40, input_dim=xtreino.shape[1]))
      mlp.add(Dropout(0.15))
      mlp.add(Dense(40,activation='relu'))
      mlp.add(Dropout(0.15))
      mlp.add(Dense(1,activation='linear'))
      mlp.compile(optimizer='adam',loss='mse', metrics=['mae'])
```

```
            mlp.fit(xtreino,y_train, epochs=1000,batch_size=16, verbose=False)
            y_pred= np.expm1(sales_scaler.
                             inverse_transform(mlp.predict(xteste).reshape(-1,1)))
        else:
            mdl[1].fit(xtreino,y_train)
            y pred=np.expm1(sales scaler.
                            inverse_transform( mdl[1].predict(xteste).reshape(-1,1)))

    end_time = time.time()
    dif = end_time-start_time

    bkp['y_pred']=y_pred
    bkp['modelo']=mdl[0]
    df_rank = pd.concat([df_rank,bkp])

    error_mae = mean_absolute_error(y_test,y_pred)
    rmse=mean_squared_error(y_test,y_pred,squared=True)
    m_rmsle = rmsle(y_test,y_pred)
    r2 = r2_score(y_test,y_pred)

    data= {'Modelo': mdl[0],
           'RMSE': rmse,
           'MAE': error_mae,
           'RMSLE': m_rmsle,
           'r2': r2,
           'periodo_treinamento':
               df2.loc[train_indice]['data'].dt.strftime('%Y-%m-%d').min()+ \
               ' a '+ df2.loc[train_indice]['data'].dt.strftime('%Y-%m-%d').max(),
           'periodo_test':  df2.loc[test_indice]['data'].dt.strftime('%Y-%m-%d').min()+ \
               ' a '+ df2.loc[test_indice]['data'].dt.strftime('%Y-%m-%d').max(),
           'DT_TRAIN_BE': DT_INI_TRAIN +'|'+DT_FIN_TRAIN,
           'DT_TESTE_BE': DT_INI_TESTE +'|'+DT_FIN_TESTE,
           'tempo_execucao': dif }


    df_metrics = df_metrics.append(data,ignore_index=True)

  dfs[mdl[0]] = df_metrics.mean()
  df_metrics.to_csv('demand_results.csv')


apply_models_price()
apply_models_demand()
```

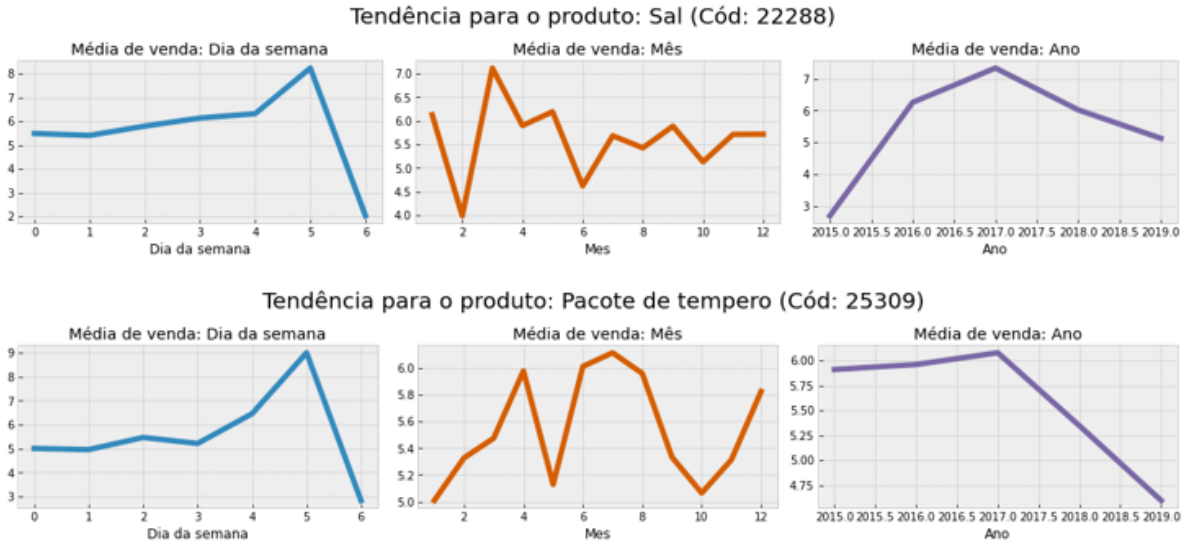
APÊNDICE B – Vendas de produtos por dia, mês e ano

Figura 54 – Tendências dos produtos de códigos 2542, 5006 e 6369.



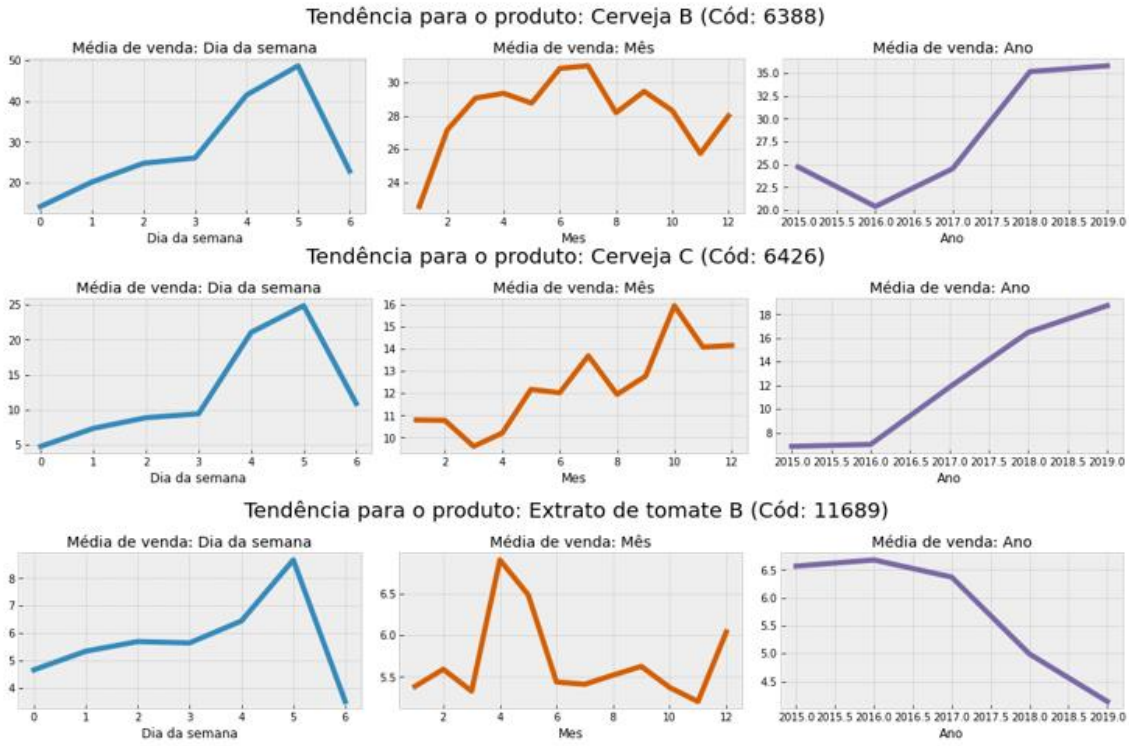
Fonte: Do autor (2021).

Figura 55 – Tendências dos produtos de códigos 22288 e 25309.



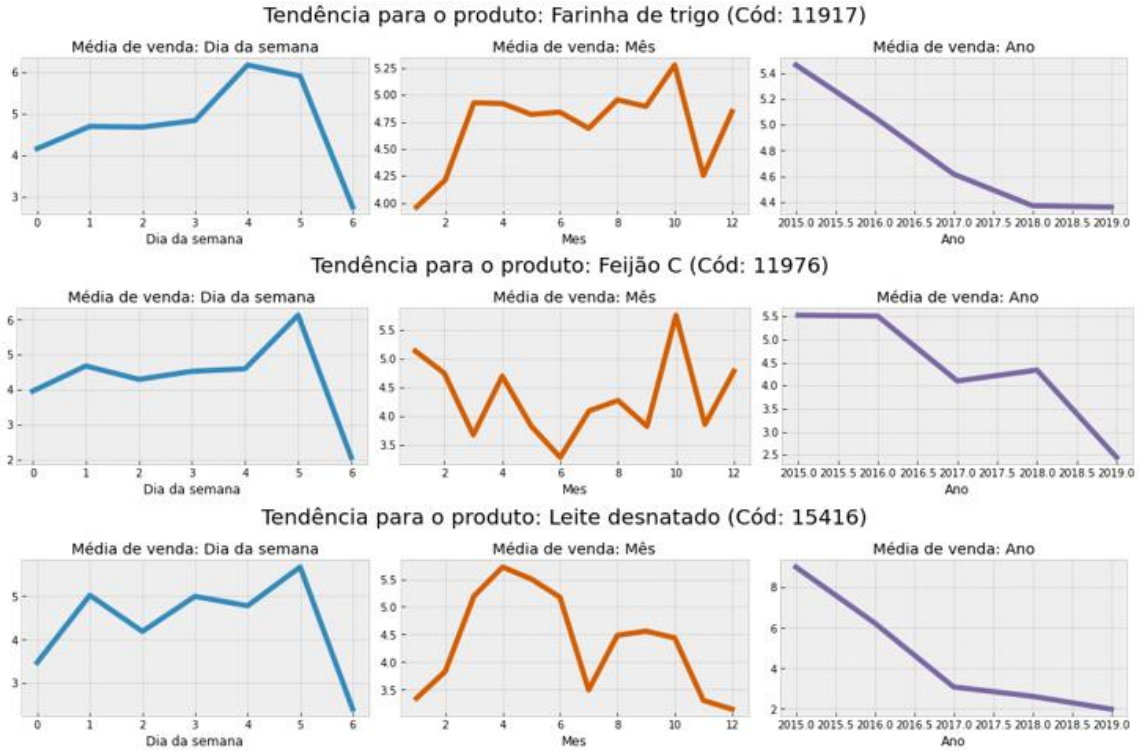
Fonte: Do autor (2021).

Figura 56 – Tendências dos produtos de códigos 6388, 6426 e 11689.



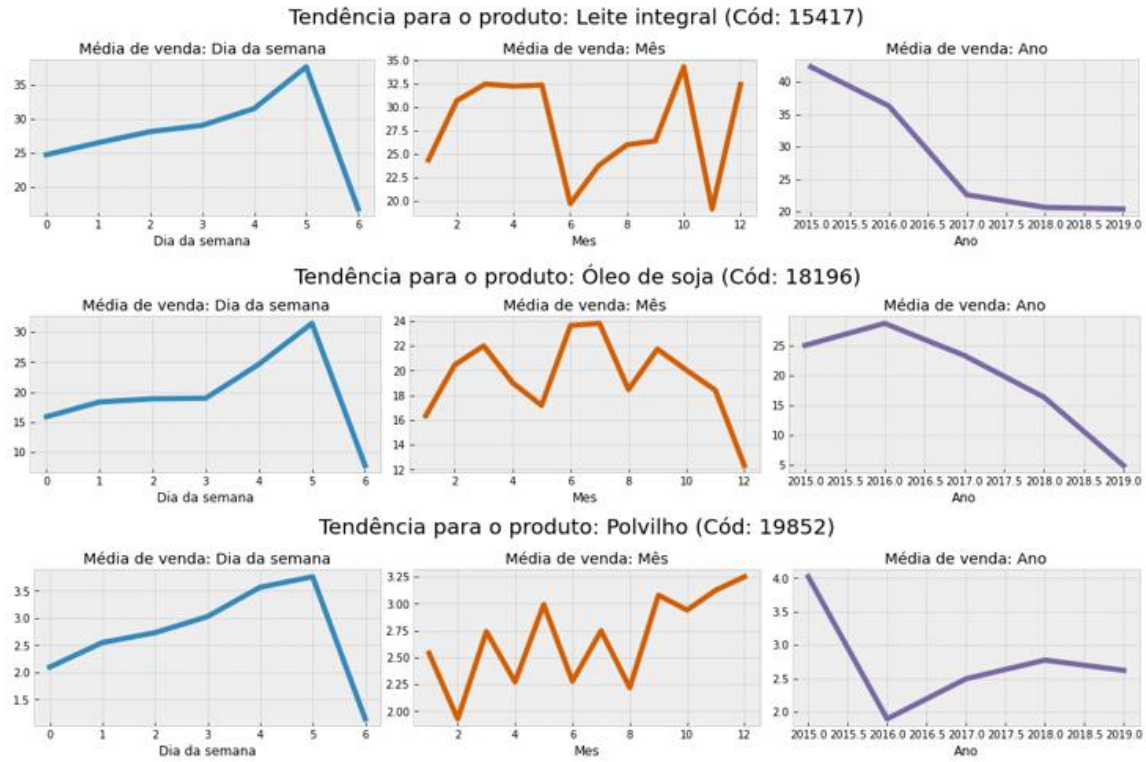
Fonte: Do autor (2021).

Figura 57 – Tendências dos produtos de códigos 11917, 11976 e 15416.



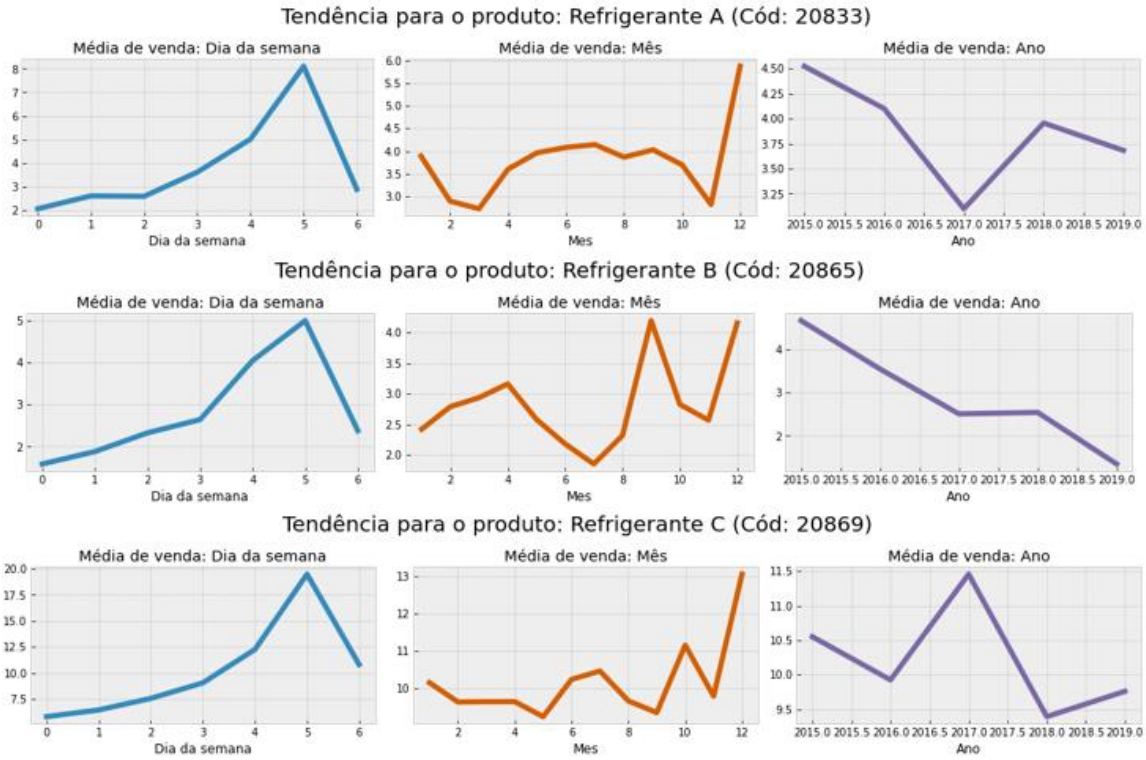
Fonte: Do autor (2021).

Figura 58 – Tendências dos produtos de códigos 15417, 18196 e 19852.



Fonte: Do autor (2021).

Figura 59 – Tendências dos produtos de códigos 20833, 20865 e 20869.



Fonte: Do autor (2021).