UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROJETO DE GRADUAÇÃO

ANDRÉ FELIPE SANTOS PEREIRA

APLICAÇÃO DE TÉCNICAS DE INTELIGÊNCIA ARTIFICIAL PARA OTIMIZAÇÃO DAS VENDAS EM E-COMMERCE

VITÓRIA-ES

DEZEMBRO/2023

André Felipe Santos Pereira

APLICAÇÃO DE TÉCNICAS DE INTELIGÊNCIA ARTIFICIAL PARA OTIMIZAÇÃO DAS VENDAS EM E-COMMERCE

Parte manuscrita do Projeto de Graduação do aluno André Felipe Santos Pereira, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Vitória-ES

Dezembro/2023

André Felipe Santos Pereira

APLICAÇÃO DE TÉCNICAS DE INTELIGÊNCIA ARTIFICIAL PARA OTIMIZAÇÃO DAS VENDAS EM E-COMMERCE

Parte manuscrita do Projeto de Graduação do aluno André Felipe Santos Pereira, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

COMISSÃO EXAMINADORA:

gov.br
Documento assinado digitalmente
HELDER ROBERTO DE OLIVEIRA ROCHA
Data: 18/12/2023 13:32:31-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Helder Roberto de Oliveira Rocha
Universidade Federal do Espírito Santo
Orientador

Prof. Dr. Jair Adriano Lima Silva
Universidade Federal do Espírito Santo
Examinador

Prof. Dr. Wesley da Silva Costa
Universidade Federal do Espírito Santo
Examinador

Vitória-ES

Dezembro/2023

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

PROTOCOLO DE ASSINATURA

O documento acima foi assinado digitalmente com senha eletrônica através do Protocolo Web, conforme Portaria UFES nº 1.269 de 30/08/2018, por
WESLEY DA SILVA COSTA - SIAPE 3358499
Departamento de Engenharia Elétrica - DEE/CT
Em 18/12/2023 às 14:41

Para verificar as assinaturas e visualizar o documento original acesse o link:
https://api.lepisma.ufes.br/arquivos-assinados/858519?tipoArquivo=O

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

PROTOCOLO DE ASSINATURA

O documento acima foi assinado digitalmente com senha eletrônica através do Protocolo Web, conforme Portaria UFES nº 1.269 de 30/08/2018, por
JAIR ADRIANO LIMA SILVA - SIAPE 1514954
Departamento de Engenharia Elétrica - DEE/CT
Em 18/12/2023 às 22:47

Para verificar as assinaturas e visualizar o documento original acesse o link:
https://api.lepisma.ufes.br/arquivos-assinados/859122?tipoArquivo=O

AGRADECIMENTOS

Gostaria de agradecer a todas essas pessoas especiais que apareceram (e estão) na minha vida e que permitem que eu continue com os meus estudos. A todos vocês, minha eterna gratidão.

Aos meus pais, pelo apoio e dedicação, não só durante toda a graduação, como durante toda a minha vida.

Aos meus irmãos por todo apoio emocional e por me servirem de inspiração.

À minha namorada por toda compreensão e apoio durante toda a graduação.

A meu orientador Prof. Dr. Helder Roberto de Oliveira Rocha por despertar meu interesse por esse tema fascinante e por toda ajuda, orientação e dedicação durante o desenvolvimento deste trabalho.

À banca examinadora pela aceitação do convite e pelo tempo investido para leitura e avaliação desse trabalho.

Agradeço à Universidade Federal do Espirito Santo pela minha formação.

RESUMO

Este trabalho insere-se na área de Inteligência Artificial (IA) aplicada ao comércio eletrônico, uma área de crescente importância dada a sua capacidade de otimizar operações, melhorar a experiência do cliente e aumentar a lucratividade. A evolução das tecnologias de IA tem permitido avanços significativos nesse setor, oferecendo soluções cada vez mais eficientes e precisas para os desafios enfrentados pelas empresas de *e-commerce*.

Apesar dos avanços, ainda existem lacunas no entendimento de como modelos de inteligência artificial podem ser otimizados para abordar desafios específicos do e-commerce, como recomendação de produtos, otimização de lucro, previsão de cancelamento de pedidos e análise de sentimentos. Há uma necessidade de pesquisa para desenvolver abordagens mais eficazes e eficientes que integrem diferentes técnicas de IA para essas aplicações.

Este trabalho propõe a utilização de aprendizado de máquina para abordar quatro problemas fundamentais em e-commerce: recomendação de produtos, otimização de lucro, previsão de cancelamento de pedidos e análise de sentimentos, que visa classificar comentários entre positivo, negativo ou neutro. O objetivo é explorar como diferentes técnicas de inteligência artificial podem contribuir para melhorar a eficiência e eficácia dessas operações.

Para a recomendação de produtos, utilizou-se a filtragem colaborativa baseada em itens. Na otimização de lucro, redes neurais foram empregadas para previsões e na sequência utilizamos o otimizador Optuna para encontrar os parâmetros de entrada do modelo que maximizam o lucro. Para a previsão de cancelamento de pedidos, recorreu-se às redes neurais com a otimização de hiperparâmetros do modelo via Optuna. Finalmente, na análise de sentimentos, foi aplicado um modelo de regressão logistica com tratamento de texto.

Os resultados demonstraram a eficácia das metodologias adotadas: (1) o sistema de recomendação apresentou uma similaridade máxima de 0.44 para o primeiro item recomendado; (2) o modelo de rede neural para otimização de lucro alcançou um R^2 de 0.98; (3) a previsão de cancelamento de pedidos obteve uma acurácia de 0.93; e (4) a análise de sentimentos alcançou uma taxa de 0.98 em acurácia.

Palavras-chave: Inteligência Artificial; E-commerce ; Redes Neurais; Otimização

ABSTRACT

This work is situated in the field of Artificial Intelligence applied to e-commerce, an area of growing importance due to its ability to optimize operations, enhance customer experience, and increase profitability. The evolution of AI technologies has enabled significant advancements in this sector, offering increasingly efficient and accurate solutions to the challenges faced by e-commerce companies.

Despite these advancements, there are still gaps in understanding how artificial intelligence models can be optimized to address specific e-commerce challenges, such as product recommendation, profit optimization, order cancellation forecasting, and sentiment analysis. There is a need for research to develop more effective and efficient approaches that integrate various AI techniques for these applications.

This work proposes the use of neural networks to tackle four fundamental problems in e-commerce: product recommendation, profit optimization, order cancellation forecasting, and sentiment analysis. The objective is to explore how different applications of neural networks can contribute to improving the efficiency and effectiveness of these operations.

For product recommendation, item-based collaborative filtering was used. In profit optimization, neural networks were employed for forecasting, with Optuna used to find the input parameters that maximize profit. For order cancellation forecasting, neural networks were used with hyperparameter optimization via Optuna. Finally, for sentiment analysis, neural networks were applied with text processing.

The results demonstrated the effectiveness of the adopted methodologies: (1) the recommendation system showed a maximum similarity of 0.44 for the first recommended item; (2) the neural network model for profit optimization achieved an R^2 of 0.98; (3) the order cancellation forecasting achieved an accuracy of 0.93; and (4) the sentiment analysis achieved an accuracy of 0.98.

Keywords: Artificial Intelligence; E-commerce; Neural Networks; Optimization

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AWS	*Amazon Web Services*
BOW	*Bag Of Words*
CMV	*Custo de Mercadoria Vendida*
FN	*Falso Negativo*
FP	*Falso Positivo*
IA	*Inteligencia Artificial*
KNN	*k-Nearest Neighbors*
ML	*Machine Learning*
PLN	*Processamento de Linguagem Natural*
R^2	*R-quadrado*
RMSE	*Root Mean Squared Error*
RNA	*Redes Neurais Artificiais*
SQL	*Structured Query Language*
SVM	*Suport Vector Machine*
SVD	*Singular Value Decomposition*
UFES	Universidade Federal do Espírito Santo
VN	*Verdadeiro Negativo*
VP	*Verdadeiro Positivo*

SUMÁRIO

1 INTRODUÇÃO

1.1 Apresentação

A última década tem sido marcada por um crescimento exponencial no número de usuários de internet e telefones celulares, impulsionado pela revolução tecnológica (MITTAL et al, 2020). Segundo dados da Fundação Getulio Vargas (FGV), em média, o Brasil possui mais de dois dispositivos digitais por habitante. Essa expansão acelerada no acesso à internet e ao uso de dispositivos móveis tem proporcionado uma gama de ferramentas de comunicação, abrindo novas possibilidades tanto para indivíduos quanto para organizações.

No contexto atual, a comunicação por meio de plataformas online desempenha um papel fundamental no marketing integrado e nas interações das empresas com seus clientes (KUMAR , 2022). A pandemia de COVID-19 acelerou ainda mais o crescimento do comércio eletrônico global, impulsionando-o a atingir a marca impressionante de US26,7 trilhões, conforme relatado pela Conferência das Nações Unidas sobre Comércio e Desenvolvimento (2021).

As principais empresas tecnológicas adotaram a inteligência artificial (IA) como se esta fosse uma das descobertas mais importantes alguma vez inventadas. Como mencionou o Chief Executive Officer (CEO) da Google, a IA pode ser comparada à descoberta do fogo e da eletricidade (PEARSON, 2020).

O uso de canais eletrônicos no marketing tem revelado resultados notáveis. Pesquisas demonstram que organizações que adotam esses canais desfrutam de taxas de conversão superiores àquelas que dependem de meios tradicionais (SAKAS et al., 2023). A personalização com base nas preferências e necessidades do cliente é fundamental para atrair e reter consumidores, contribuindo para a satisfação do cliente e a fidelização à marca (TONG et al., 2014).

A aplicação de Inteligência Artificial (IA) no e-commerce está revolucionando a maneira como os clientes interagem com as lojas online, proporcionando uma experiência de compra altamente personalizada e interativa. Este avanço contribui significativamente para aprimorar o ciclo de vida do cliente. Nesse contexto, o objetivo deste projeto é empregar técnicas avançadas de aprendizado de máquina para mapear e otimizar cada etapa da jornada do cliente no e-commerce.

1.2 Objetivos

Objetivo Geral

Este trabalho tem como objetivo geral o desenvolvendo de modelos de redes neurais para otimizar o ciclo de compras em um e-commerce. Para isso, pretende-se modelar um sistema de recomendação de produtos, maximizar o lucro, provisionar cancelamentos e análise de sentimentos classificando comentários em produtos. Isso resultará em uma experiência mais personalizada, preços competitivos, redução de cancelamentos e insights valiosos dos clientes.

Objetivos Específicos

Os objetivos específicos deste projeto abrangem a extração, pré-processamento e tratamento de dados do banco de dados, bem como a aplicação de técnicas de recomendação, regressão, classificação e processamento de linguagem natural e, por fim, a análise dos resultados. Para alcançar o objetivo geral deste trabalho, os seguintes objetivos específicos deverão ser atendidos:

- Realizar a extração dos dados nos bancos de dados;
- Realizar a tratativa dos dados;
- Realizar o treinamento de modelos de machine learning em cada um dos casos;
  - Sistema de Recomendação;
  - Regressão: Precificação otimizada;
  - Classificação: Previsão de cancelamento de pedidos;
  - Processamento de Linguagem Natural: Análise de Sentimentos.
- Validar e testar desempenho de modelos.

1.3 Estrutura do Texto

O presente trabalho é dividido em 5 capítulos, descritos abaixo:

- **Introdução**: este capítulo tem como objetivo contextualizar o tema deste trabalho, apresentando suas justificativas, trabalhos relacionados e objetivos;
- **Referencial Teórico**: este capítulo dedica-se a explicar a fundamentação teórica necessária para a realização deste trabalho;
- **Metodologia**: este capítulo é dedicado à apresentação da solução proposta para o problema em estudo;
- **Simulações e Resultados**: neste capítulo são apresentadas as métricas de desempenho e os resultados obtidos após os experimentos;
- **Conclusões**: no capítulo final deste trabalho são apresentadas as conclusões e os trabalhos futuros.

2 REFERENCIAL TEÓRICO

2.1 Aprendizado de Máquina

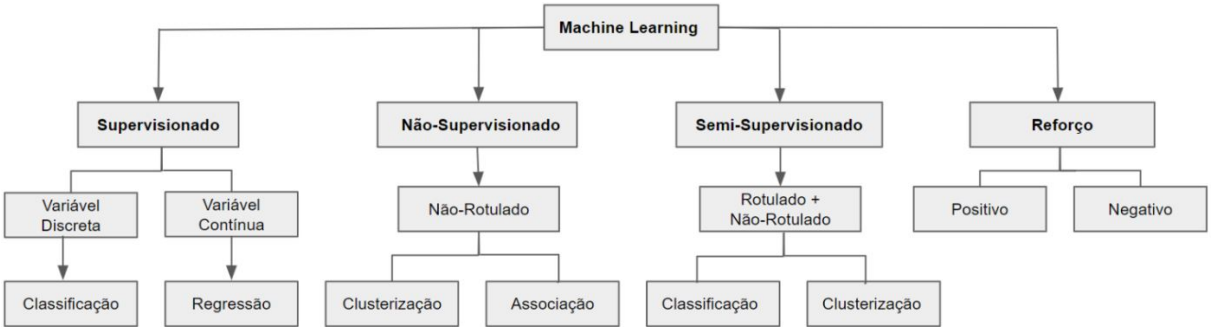
O Aprendizado de Máquina tem suas raízes nos princípios fundamentais da estatística, cálculo matemático e programação de computadores. Por meio desses conceitos, é possível treinar um computador para fazer previsões precisas de eventos previamente desconhecidos, desde que existam dados históricos capazes de descrever tais ocorrências (TAULLI, 2019). A essência do Aprendizado de Máquina é capacitar máquinas a generalizarem a partir dos dados disponíveis, permitindo-lhes tomar decisões para novos dados (BISHOP, 2006).

O processo de Aprendizado de Máquina geralmente envolve várias etapas fundamentais:

- Coleta de Dados: Reunir e preparar os dados relevantes necessários para treinar o modelo.
- Pré-processamento de Dados: Limpar, normalizar e transformar os dados em um formato adequado para o treinamento.
- Treinamento do Modelo: Utilizar algoritmos e técnicas de aprendizagem para ajustar o modelo com os dados de treinamento, permitindo que ele aprenda com os padrões.
- Avaliação do Modelo: Verificar a precisão e o desempenho do modelo usando dados de teste ou validação.
- Ajuste e Otimização: Realizar ajustes no modelo para melhorar seu desempenho e generalização.
- Implantação e Monitoramento: Implementar o modelo em ambiente real e monitorar seu desempenho contínuo (BISHOP, 2006).

Existem quatro tipos principais de Aprendizado de Máquina: Supervisionado, Não Supervisionado, Semi-Supervisionado e por Reforço, conforme apresentado na Figura 1. Neste projeto abordaremos com mais detalhes o aprendizado supervisionado.

Figura 1 – Tipos de aprendizados com suas propriedades e aplicações.



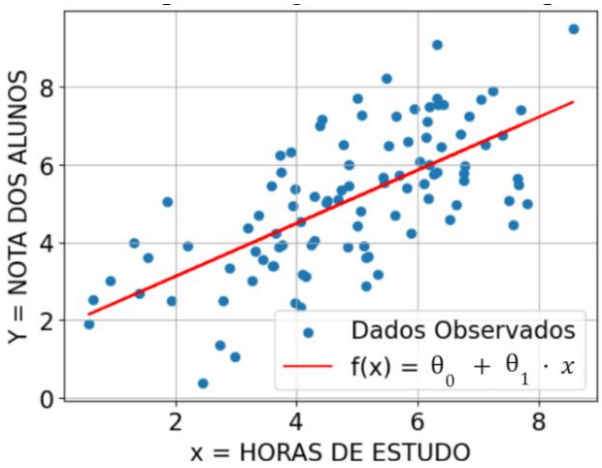
Fonte: (Sarker, 2021)

Nota: Modificado pelo autor

2.1.1 Aprendizado supervisionado

É usado quando temos dados rotulados, e o objetivo é prever ou classificar novos exemplos com base em exemplos anteriores. Para isso, utiliza-se um algoritmo que observa alguns exemplos de pares de entrada e saída, e aprende uma função que faz o mapeamento da entrada para a saída (RUSSEL; NORVIG, 2009). Conforme apresentado na Figura 2, um modelo de regressão linear é representado por uma função $f(x) = \theta_0 + \theta_1 \times x$, e durante a fase de treinamento o modelo irá ajustar os valores de θ_0 e θ_1, chamados de parâmetros da função, para criar uma reta que correlacione as duas variáveis x e $f(x)$, em que representam os dados de entrada e a variável de saída, respectivamente.

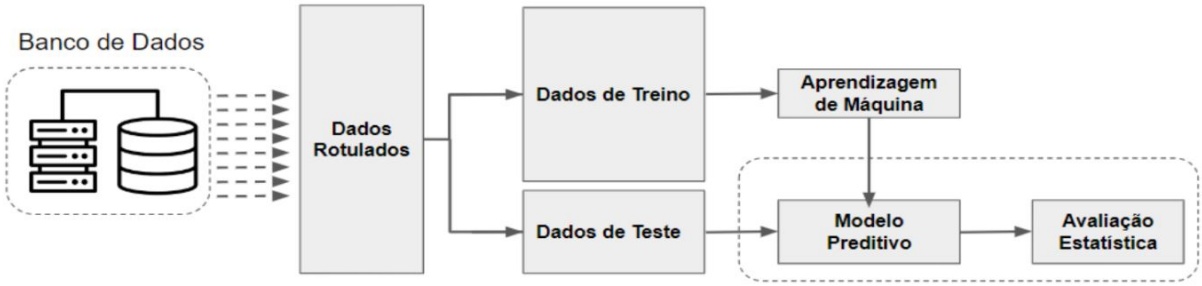
Figura 2 – Exemplo de regressão linear simples.



Fonte: O próprio autor

Dessa forma, ao adicionar novos dados ao sistema ou mesmo os dados de teste, a função conseguirá prever o valor esperado, vale ressaltar que será carregando consigo uma taxa de erro. Em machine learning não existe erro zero e uma das chaves do aprendizado supervisionado é que deve haver grandes quantidades de dados (TAULLI, 2019).

Figura 3 – Modelo de aprendizado supervisionado.



Fonte: O próprio autor

Dentro do processo de aprendizado supervisionado um conjunto de dados coletados que já foi pré-processado, é separado em dados de treino e de teste, conforme apresentado na Figura 3. Com os dados de treino são ajustados os parâmetros do modelo de machine learning. Com os dados de teste, desconhecidos até então pelo nosso modelo treinado, são realizadas novas previsões e captadas as estatísticas de predição do nosso modelo. Desta maneira, verifica- se a viabilidade da produção e operação de previsões(RUSSEL; NORVIG, 2009). No aprendizado supervisionado existe uma variável *target* no qual podemos comparar a eficiência do nosso modelo.

2.2 Funcionamento do Aprendizado de Máquina

2.2.1 Generalização

Como vimos até agora, o aprendizado de máquina é uma excelente ferramenta para se fazer previsões de eventos futuros nas mais diversas áreas. Para isso, utilizamos dados históricos para treinar um modelo que explique um determinado evento ou valor. No entanto, não queremos apenas que a o aprendizado obtenha os exemplos de treinamento corretos, também queremos que generalize para novas instâncias não apresentadas ao modelo anteriormente (GROSSE, 2023). Dizemos que uuma hipótese é generalizada se for possível prever corretamente o valor de saída da função para novos exemplos de entrada (RUSSEL; NORVIG, 2009). Componentes do aprendizado, considerando uma regressão linear simples:

- Input:

$$x \tag{2.1}$$

- Output:

$$y = 0 + 1x \tag{2.2}$$

- Função objetivo:

$$f : x \to y \tag{2.3}$$

- Espaço de hipóteses:

$$h : x \to y \tag{2.4}$$

- Função ótima:

$$g \approx f \tag{2.5}$$

A abordagem clássica do aprendizado é fundamentada na suposição de que temos um espaço de hipóteses, representado por h, que contém funções capazes de se aproximar da função objetivo (MURPHY, 2012). À medida que aumentamos o número de exemplos de treinamento, a função ótima tende a ser atingida e nossa confiança no conceito a ser aprendido cresce (MITCHELL, 1997). O processo de treinamento consiste em explorar o espaço de hipóteses, ajustando os parâmetros de aprendizado, a fim de encontrar a melhor aproximação da função objetivo. E a melhor aproximação é aquela que apresenta a menor taxa de erros ao realizar previsões em novos dados.

A formulação matemática do espaço de hipóteses é definida por:

$$\sum_{i=1}^{d} w_i \cdot x_i \tag{2.6}$$

$$h(x) = \text{sign}\left(\left(\sum_{i=1}^{d} w_i \cdot x_i\right) - \text{threshold}\right) \tag{2.7}$$

Cada hipótese é representada pela Equação 2.6, e w representa o coeficiente atribuído aos dados de entrada, ou também conhecido como vetor de pesos. As diferentes combinações de pesos w e o threshold formarão hipóteses distintas. Essa formulação matemática é fundamental para o processo de aprendizado, pois permite representar diferentes possibilidades de modelos que se ajustam aos dados de treinamento e, assim, encontrar a melhor aproximação da função objetivo.

2.2.2 Função de Custo (*Cost Function*)

A função de custo, também conhecida como função de perda, é um componente fundamental no campo do aprendizado de máquina e otimização. Essa função descreve a diferença entre

as previsões feitas por um modelo e os valores reais dos dados. Dessa forma, a função de custo é essencial para a avaliação do desempenho do modelo e para o processo de treinamento, uma vez que guia a busca pelos melhores parâmetros através de métodos de otimização como o gradiente descendente (HASTIE; TIBSHIRANI; FRIEDMAN, 2009). A formulação matemática para a função de custo:

- Hipóteses:

$$h_0(x) = \Theta_0 + \Theta_1 \times x \tag{2.8}$$

- Parâmetros:

$$\Theta_0, \Theta_1 \tag{2.9}$$

- Função de custo:

$$J(\Theta_0, \Theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\Theta(x^{(i)}) - y^{(i)})^2 \tag{2.10}$$

- Gradiente Descendente:

$$\min(J(\Theta_0, \Theta_1)) \tag{2.11}$$

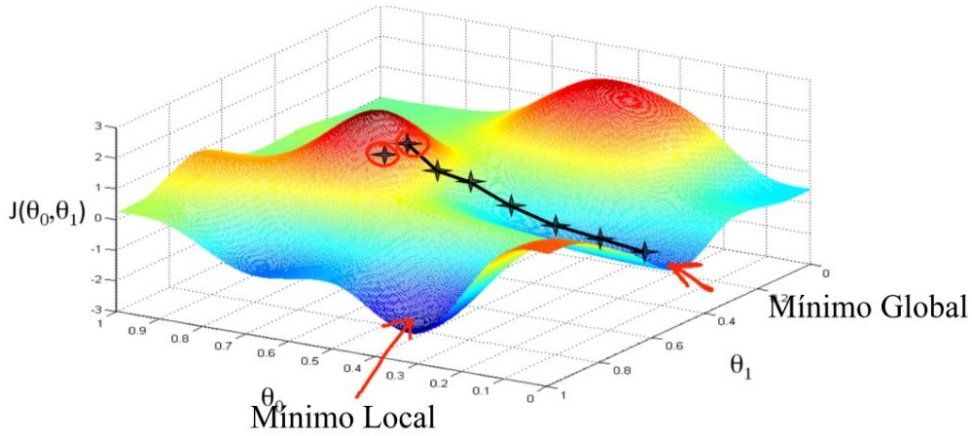
Comparando uma previsão contra o seu valor real, usando uma função de custo, determinamos o nível de erro do modelo. Por ser uma formulação matemática, a função de custo expressa o nível de erro em forma numérica. Utilizamos o resultado da função de custo para atualizar os parâmetros para que na iteração seguinte a função de hipótese se aproxime da função objetivo, isso é feito por meio do algoritmo do Gradiente Descendente.

2.2.3 Gradiente Descendente - Minimizando a Função de Custo

Uma vez que a função de custo é definida, podemos utilizar o algoritmo do gradiente descendente para minimizar o erro, ou seja, encontrar o valor mínimo global da função. Inicialmente, escolhemos um ponto de partida no espaço de parâmetros, em seguida, nos movemos para um ponto vizinho, utilizando passos que estão em direção ao declive da função. Repetimos esse processo iterativamente até convergir para a mínima perda possível (RUSSEL; NORVIG, 2009).

O tamanho do passo, conhecido como taxa de aprendizagem, é um parâmetro essencial quando tentamos minimizar a perda em um problema de aprendizagem. Essa taxa pode ser uma constante fixa ou pode diminuir ao longo do tempo à medida que o processo de aprendizagem avança (RUSSEL; NORVIG, 2009).

Figura 4 – Gráfico 3D da função de custo.



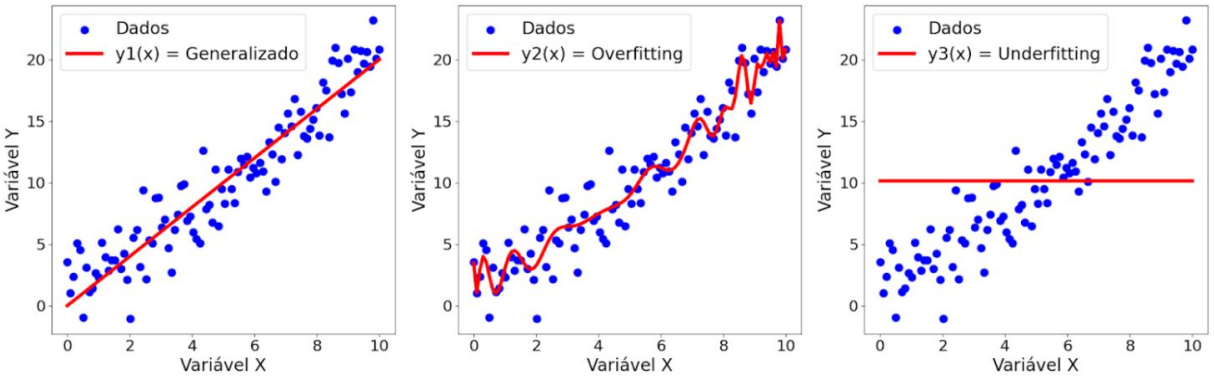
Dada uma função de custo $J(\Theta_0, \Theta_1)$ em que Θ_0, Θ_1 são os parâmetros da função, quando melhor os valores de parâmetros, menor será o valor de J e chegaremos ao mínimo global.

2.2.4 Underfitting e Overfitting

O *overfitting* ocorre quando um modelo é excessivamente ajustado aos dados de treinamento, tornando-se muito complexo e capaz de capturar até mesmo os ruídos e variações aleatórias dos dados, conforme a Figura 5(b). Isso pode levar a uma perda de capacidade de generalização, resultando em um desempenho insatisfatório quando o modelo é aplicado a novos dados (BISHOP, 2006). Para evitar o *overfitting*, é crucial empregar técnicas como a regularização, redução de complexidade do modelo ou aumento do tamanho do conjunto de treinamento (HASTIE; TIBSHIRANI; FRIEDMAN, 2009). Por outro lado, o *underfitting* ocorre quando o modelo é muito simples para capturar os padrões fundamentais presentes nos dados. Nesse caso, o modelo falha em se ajustar adequadamente aos dados de treinamento, resultando em uma capacidade de generalização insuficiente, conforme a Figura 5(c). O *underfitting* pode ser mitigado por meio do aumento da complexidade do modelo, usando técnicas mais avançadas de aprendizado de máquina ou aumentando a quantidade de dados de treinamento disponíveis (HASTIE; TIBSHIRANI; FRIEDMAN, 2009).

Portanto, para construir modelos de aprendizado de máquina eficazes, é essencial encontrar um equilíbrio entre o *overfitting* e o *underfitting*, ajustando a complexidade do modelo e a quantidade de dados de treinamento de acordo com a natureza do problema em questão, conforme a Figura 5(a). Essas estratégias permitem que o modelo seja capaz de capturar os padrões relevantes dos dados, ao mesmo tempo que generaliza bem para novos exemplos não vistos, garantindo, assim, um melhor desempenho em aplicações do mundo real.

Figura 5 – Exemplo de regressão linear para um modelo (a) Modelo Gereralizado, (b) Modelo sofrendo *overfitting* e (c) Modelo sofrendo *underfiting*.



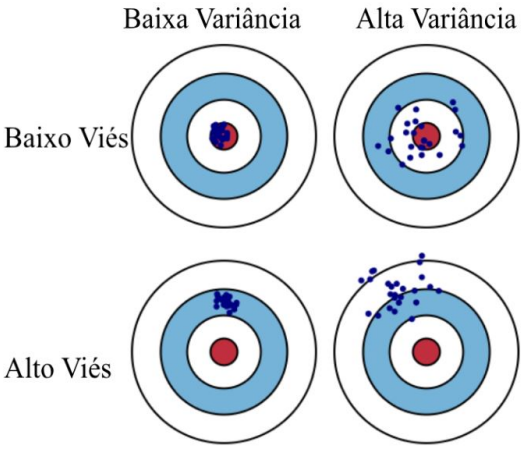
Fonte: O próprio autor

2.2.5 Viés e Variância

O viés e a variância são conceitos fundamentais no aprendizado de máquina, desempenhando um papel crucial no desempenho e generalização dos modelos. O viés refere-se à simplificação excessiva que um modelo faz sobre os dados, resultando em subestimação da complexidade e dificuldades de ajuste aos dados de treinamento (MITCHELL, 1997).

Já a variância está relacionada à sensibilidade do modelo a pequenas variações nos dados, levando a *overfitting* quando o modelo se ajusta muito bem aos dados de treinamento, mas falha em generalizar para novos dados. Desejamos encontrar o equilíbrio entre baixo viés e baixa variância para que tenhamos um modelo mais assertivo, como ilustrado na Figura 6. Dessa forma minimizamos o erro total e maximizamos a capacidade de generalização (BISHOP, 2006).

Figura 6 – Ilustração gráfica de viés e variância.

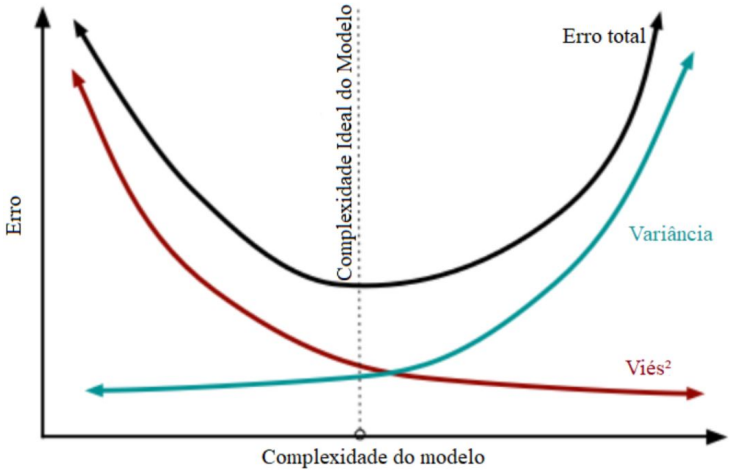


Fonte: (Fortmann-Roe, 2012).

Nota: Traduzido pelo autor

Para isso, é essencial utilizar técnicas como validação cruzada, regularização e seleção de recursos para encontrar o modelo mais equilibrado que atenda aos requisitos específicos do problema em questão. Uma das variáveis que impacta o viés e a variância do modelo é a complexidade do modelo, apresentado na Figura 7. Em muitos casos, a complexidade do modelo está relacionada ao número de parâmetros que estamos usando para aproximar a função objetivo (FORTMANN-ROE, 2012). E embora o uso de um modelo mais complicado sempre reduza o viés, em algum ponto o modelo se torna muito complexo para a quantidade de dados de treinamento e o erro de generalização se torna grande devido à alta variância(FORTMANN-ROE, 2012).

Figura 7 – Viés e variância contribuindo para o erro total.



Fonte: (Fortmann-Roe, 2012).

Nota: Traduzido pelo autor

2.3 Redes Neurais

As redes neurais artificiais ou RNA são métodos populares no campo do aprendizado de máquina, projetados para emular os processos de aprendizado observados em organismos vivos. No sistema nervoso humano, há estruturas conhecidas como neurônios, que desempenham um papel fundamental (AGGARWAL, 2018). Segundo Mitchell (1997) as redes neurais artificiais estão entre os métodos de aprendizado mais eficazes atualmente conhecidos.

Os neurônios interagem através de conexões chamadas axônios e dendritos, formando sinapses (HAYKIN, 2009). Cada entrada para um neurônio é dimensionada com um peso, que afeta a função calculada naquela unidade. Uma rede neural é um processador massivamente paralelo distribuído composto de unidades de processamento simples, que tem uma propensão natural para armazenar conhecimento experiencial e disponibilizá-lo para uso, assemelha-se ao cérebro em dois aspectos:

- O conhecimento é adquirido pela rede a partir de sua formação por meio de um processo de aprendizagem.
- As forças de conexão interneuronais , conhecidas como pesos sinápticos, são usadas para armazenar o conhecimento adquirido (HAYKIN, 2009),

É importante notar que as conexões sinápticas podem se fortalecer ou enfraquecer em resposta a estímulos externos, contribuindo para o processo de aprendizado em seres vivos (AGGARWAL, 2018). As sinapses, ou terminações nervosas, são unidades estruturais e funcionais elementares que medem as interações entre os neurônios(HAYKIN, 2009). Segundo Mitchell (1997), estima-se que o cérebro humano, por exemplo, contenha uma rede densamente interconectada de aproximadamente 1011 neurônios, cada um conectado, em média, a 104 outros e além disso, a atividade do neurônio é tipicamente excitada ou inibida por meio de conexões com outros neurônios.

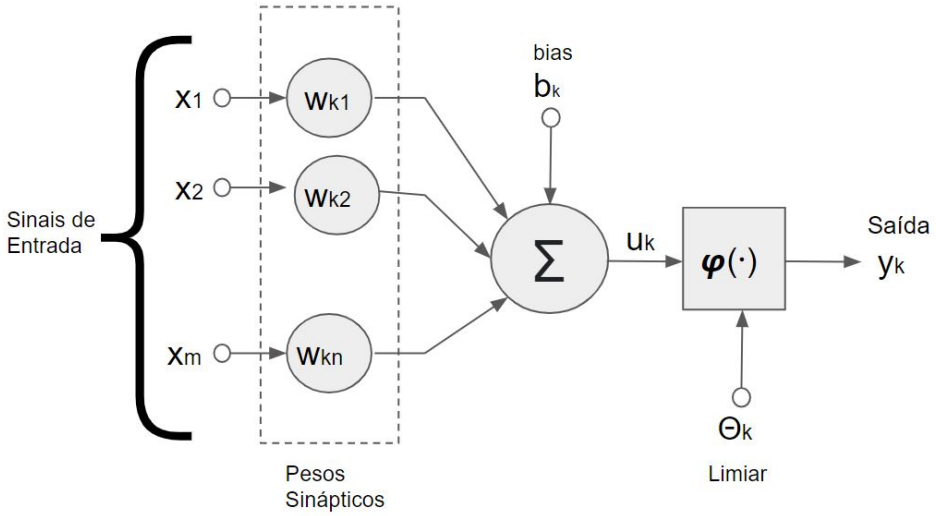
Uma rede neural artificial calcula uma função das entradas propagando os valores computados dos neurônios de entrada para o(s) neurônio(s) de saída e usando os pesos como parâmetros intermediários. A aprendizagem ocorre alterando os pesos que conectam os neurônios (AGGARWAL, 2018), como poderemos ver a seguir.

A arquitetura básica das redes neurais é mostrada no Figura 8, onde x_1, x_2, ..., x_m são os sinais de entrada; W1, W2, ..., Wn são os respectivos pesos sinápticos do neurônio k; "uk"é a saída do combinador linear devido aos sinais de entrada; θ é a função de ativação; e yk é o sinal de saída do neurônio.. Posteriormente, a função Limiar é aplicada para converter o valor agregado em um rótulo de classe. A função de sinal serve como uma função de ativação (AGGARWAL, 2018). Na rede de camada única, um conjunto de entradas é mapeado diretamente para uma saída usando uma variação generalizada de uma função linear. Essa instanciação simples de uma rede neural também é chamada de perceptron.

Nas redes neurais multicamadas, os neurônios são organizados em camadas, nas quais as camadas de entrada e saída são separadas por um grupo de camadas ocultas. Essa arquitetura em camadas da rede neural também é chamada de rede *feed-forward* (AGGARWAL, 2018), como veremos a seguir. Até agora, identificamos três elementos básicos do modelo neural:

- Um conjunto de sinapses, ou elos de conexão, cada um dos quais é caracterizado por um peso;
- Um somador para soma dos sinais de entrada, ponderados pelas respectivas forças sinápticas do neurônio;

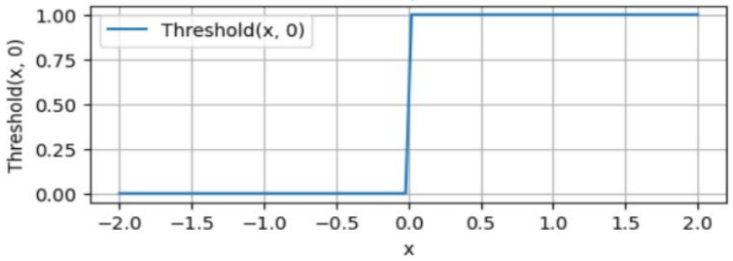
Figura 8 – Modelo de um Neurônio Artificial



Fonte: Adaptado de HAYKIN, Simon.

- A função de ativação, na medida em que comprime (limita) a faixa de amplitude permitida do sinal de saída para algum valor finito (HAYKIN, 2009).

Figura 9 – Curva da função limiar ou função threshould



Fonte: O prório autor

Uma função de ativação para limitar a amplitude da saída de um neurônio. A função de ativação também é chamada de função de esmagamento, na medida em que reduz (limita) a faixa de amplitude permitida do sinal de saída a algum valor finito (HAYKIN, 2009). A função de limiar, ou *Threshold Function* em inglês, é uma função matemática que toma um valor de entrada e retorna um valor de saída binário (0 ou 1) com base em um valor limite pré-definido, conforme apresentado na Figura 9. Se o valor de entrada ultrapassar o limite, a função retorna 1; caso contrário, retorna 0. Essencialmente, a função de limiar age como um interruptor, ativando-se ou desativando-se dependendo se a entrada excede ou não o valor de limiar.

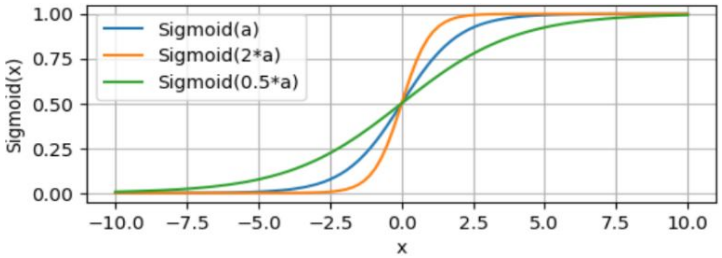
A função de limiar pode ser formalmente definida da seguinte forma:

- $$f(x) = 1, \text{ se } x \geq \text{limite} \tag{2.12}$$

- $$f(x) = 1, \text{ se } x \leq \text{limite} \tag{2.13}$$

A função sigmóide, também conhecida como função logística, é uma função matemática que transforma um valor de entrada em um valor no intervalo entre 0 e 1. Ela é amplamente utilizada em diversas áreas, incluindo em redes neurais, onde costumava ser uma função de ativação comum em camadas intermediárias.

Figura 10 – Curvas da função Sigmoid.



Fonte: O prório autor

A função sigmóide é definida matematicamente como:

$$\sigma(v) = \frac{1}{1 + e^{-av}} \tag{2.14}$$

onde a é o parâmetro de inclinação da função sigmóide. Variando o parâmetro a, obtemos funções sigmóides de diferentes inclinações, conforme ilustrado na Figura 10. No limite, conforme o parâmetro de inclinação se aproxima do infinito, a função sigmóide torna-se simplesmente uma função de limite.

A principal diferença entre a função sigmoide e a função limiar é a natureza de suas saídas e suas propriedades matemáticas. Enquanto a sigmóide gera saídas contínuas no intervalo entre 0 e 1, a função limiar produz saídas binárias (0 ou 1) com base em um valor de limiar. Além disso, a função sigmóide é diferenciável em todos os pontos, permitindo o uso de técnicas de otimização, enquanto a função limiar não é diferenciável em pontos específicos, o que pode limitar sua aplicação em redes neurais e treinamento baseado em gradientes.

Ainda existem ainda outras funções de ativação como Softmax e ReLU, cada uma dessas funções tem características específicas que as tornam mais adequadas para certos tipos

de redes neurais e problemas. A escolha da função de ativação apropriada é um aspecto importante no desenho de uma rede neural eficaz.

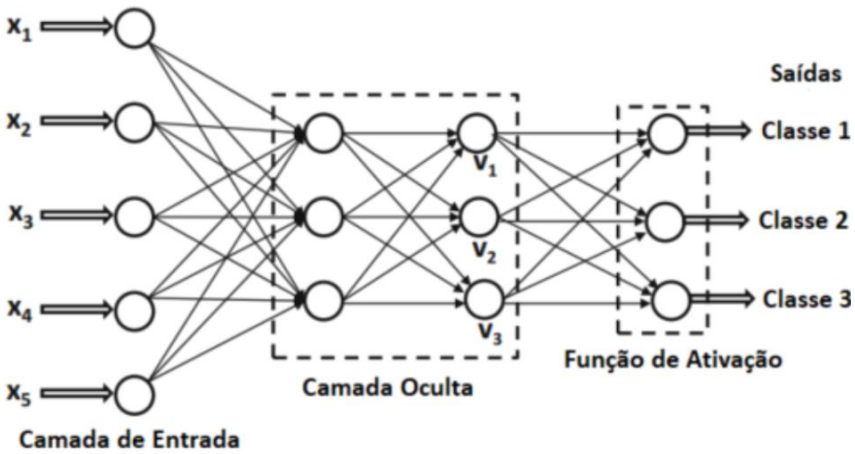
2.3.1 Feed-Propagation

O termo *feedforward propagation* (propagação direta) refere-se à etapa de encaminhar dados de entrada através de uma rede neural para calcular a saída correspondente. Isso envolve a passagem dos dados da camada de entrada para a camada de saída através das camadas intermediárias, aplicando multiplicação de pesos e funções de ativação. O algoritmo *feed-propagation* é a técnica de aprendizado de RNA mais comumente usada. É apropriado para problemas com as seguintes características:

- A saída da função de destino pode ser de valor discreto, valor real ou um vetor de vários atributos de valor real ou discreto.
- Os exemplos de treinamento podem conter erros.
- Longos tempos de treinamento são aceitáveis.
- Pode ser necessária uma avaliação rápida da função alvo aprendida (MITCHELL, 1997).

As instâncias são representadas por muitos pares de valor de atributo. Durante a fase de *Forward Propagation*, os dados fluem da camada de entrada para a camada de saída, passando pelas camadas intermediárias, como apresentado na Figura 11.

Figura 11 – Rede Neural Multicamadas Genérico



Fonte: (AGGARWAL, 2018)

À medida que os dados passam pelas camadas, cada neurônio realiza um cálculo baseado em suas entradas e pesos, aplicando uma função de ativação para determinar sua saída. Essa sequência de cálculos transforma os dados brutos em informações significativas e, finalmente, em previsões ou classificações.

O processo de *backpropagation*, ou retropropagação do erro, é a chave para ajustar os pesos das conexões entre os neurônios, permitindo que a rede neural melhore seu desempenho ao longo do tempo(TAULLI, 2019).Segundo Mitchell (1997) o algoritmo back propagation assume que a rede é uma estrutura fixa que corresponde a um grafo direcionado, possivelmente contendo ciclos.

Durante o *backpropagation*, a diferença entre a saída real da rede e a saída desejada (o erro) é calculada. Essa diferença é então propagada de volta através das camadas da rede, determinando como os pesos devem ser ajustados para minimizar o erro. Isso é feito usando um algoritmo de otimização, como o gradiente descendente, que ajusta gradualmente os pesos para melhorar a precisão das previsões.

2.4 Sistema de recomendação

Os sistemas de recomendação tornaram-se uma parte essencial do ecossistema digital nos últimos anos, facilitando a navegação dos usuários em vastos espaços de informação e ajudando empresas a oferecer produtos e serviços mais relevantes a seus clientes. Segundo Sharma e Gera (2013), um sistema de recomendação é um sistema inteligente que sugere itens aos usuários que possam ser de seu interesse. Estes sistemas são empregados em várias plataformas, desde e-commerce, como Amazon.com e Alibaba à plataforma de streaming como a Netflix.

Após a implementação de sistemas de recomendação de produtos, diversas empresas observaram um aumento significativo em suas vendas. Por exemplo, a Amazon registrou um crescimento de aproximadamente 29% em suas vendas no segundo trimestre fiscal depois de adotar esse sistema (MANGALINDAN, 2012). De forma similar, após o anúncio de um novo sistema de publicidade baseado em recomendações, o Alibaba, uma das gigantes do comércio eletrônico, viu suas ações subirem em 51% em um período de três meses (BLOOMBERG, 2019). Esses dados evidenciam a importância e o impacto dos sistemas de recomendação no setor de e-commerce, especialmente à medida que as empresas buscam oferecer uma experiência mais personalizada aos seus clientes.

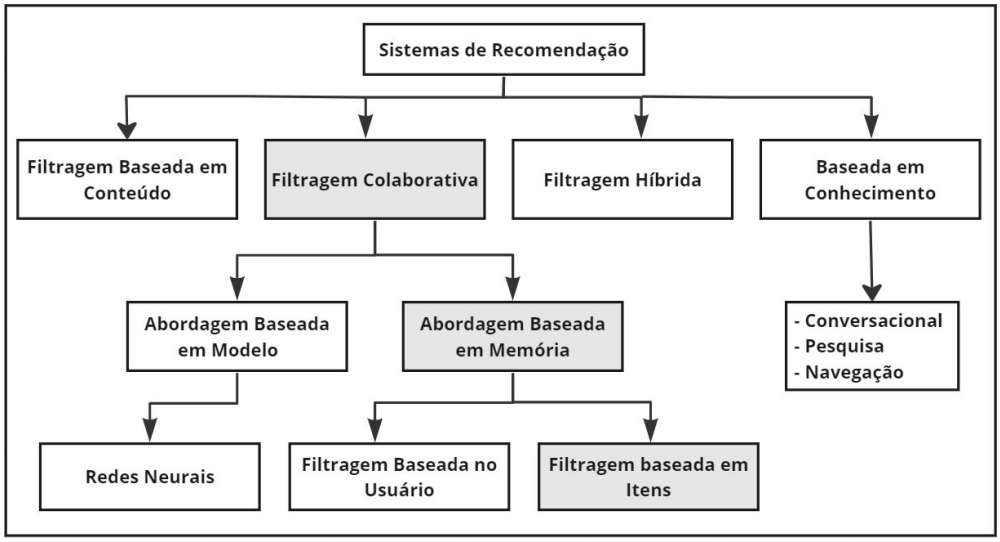
Em sistemas de recomendação, as duas principais entidades são os "Usuários"e os "Itens",

que a base do sistema e podem ser formalmente definidas para representar, respectivamente, os indivíduos que usam o sistema e os itens que podem ser recomendados a eles(KEIKHOSROKIANI; FYE, 2023).

Estas entidades podem ser formalmente definidas da seguinte maneira:

- Conjunto de Usuários (C): Representa todos os usuários que fazem parte do sistema.
- Conjunto de Itens (S): Engloba todos os itens possíveis que podem ser recomendados aos usuários.
- Função de Utilidade (U): Esta função mede a utilidade de um item específico s pertencente ao conjunto S para um usuário c do conjunto C (KEIKHOSROKIANI; FYE, 2023).

Figura 12 – Hierarquia do sistema de recomendação baseado em filtragem.

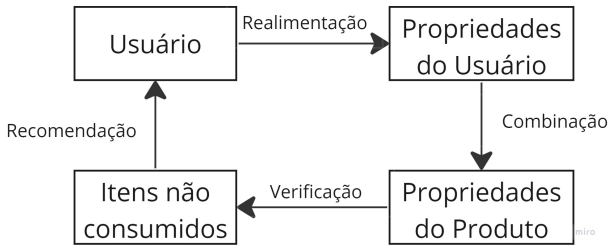


Fonte: (KEIKHOSROKIANI; FYE, 2023).

Os algoritmos de recomendação baseados em conteúdo focam em sugerir itens ou produtos que se alinham ao histórico de preferências do usuário ou à sua busca atual. A essência dessa abordagem é agrupar os itens de acordo com sua correspondência ao perfil do usuário-alvo. Especificamente, a semelhança entre a descrição do item e o perfil do usuário é calculada, e os itens que melhor se alinham a esse perfil são recomendados(ALAMDARI et al., 2020), como ilustrado na Figura 13.

Um sistema de filtragem colaborativa representa usuários com suas classificações em um conjunto de itens chamados de vetores de classificação e então faz previsões sobre preferências desconhecidas dos usuários ao construir uma matriz que mapeia suas escolhas ou afinidades por diferentes itens. Depende diretamente dos dados de treinamento

Figura 13 – Funcionamento de filragem baseada em conteúdo.



Fonte: Adaptado de (SHARMA; GERA, 2013)

armazenados e faz previsões com base na semelhança ou proximidade com esses dados. Não generaliza além dos dados armazenados, dessa forma, apresenta um ponto negativo chamado *cold start* ou início frio, que será detalhado ao longo deste capítulo.

Existe também a abordagem híbrida, que une diversos métodos para uma recomendação mais refinada. A eficácia dos sistemas de recomendação pode ser aumentada pela fusão das metodologias de filtragem colaborativa e baseadas em conteúdo, aproveitando suas vantagens complementares. Estudos variados têm demonstrado melhorias significativas no desempenho dos sistemas de recomendação híbridos (SHARMA; GERA, 2013).

Por fim, os sistemas que se fundamentam em conhecimento especializado utilizam insights profundos sobre como os atributos específicos dos itens atendem às necessidades e preferências dos usuários.

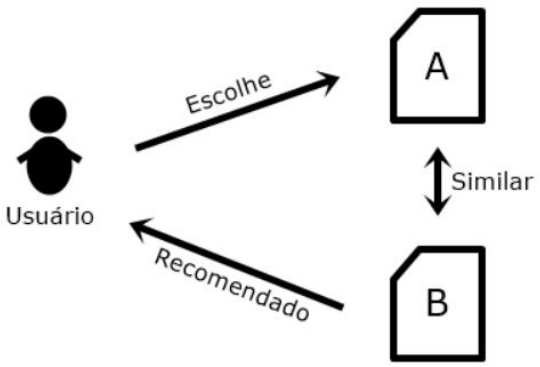
2.4.1 Filtragem Baseado em Colaboração

Os sistemas baseados em Filtragem Colaborativa operam coletando opiniões dos usuários, geralmente na forma de avaliações, para itens específicos em um campo de interesse (LAKSHMI; BHAVANI, 2017). Eles identificam padrões de avaliações similares entre os usuários para fundamentar suas recomendações (SHARMA; GERA, 2013). Essencialmente, tais sistemas fazem recomendações a um usuário levando em consideração o que outros usuários com gostos parecidos preferiram.

Na Figura14, é ilustrado o funcionamento de um sistema de recomendação baseado em conteúdo. Neste modelo, quando um usuário seleciona um item específico, denominado "A", o sistema deve ser capaz de identificar outro item, "B", que possui características similares ao item "A"e, em seguida, sugerir esse item ao usuário(ROLIM et al., 2017).

A Filtragem Colaborativa destaca-se principalmente por sua capacidade de personalização,

Figura 14 – Funcionamento de filtragem colaborativa baseada em itens.



Fonte: (ROLIM et al., 2017)

adaptando-se às mudanças nas preferências dos usuários e oferecendo uma diversidade de recomendações ao analisar o comportamento de usuários similares (KEIKHOSROKIANI; FYE, 2023). Contudo, enfrenta desafios significativos como o problema de inicialização a frio, no qual novos usuários ou itens carecem de dados históricos para formar recomendações confiáveis. Por exemplo, um novo cliente sem histórico de avaliação pode não encontrar um padrão de produtos semelhante, ou um novo produto pode não ser recomendado por não ter um histórico de avaliações (SU; KHOSHGOFTAAR, 2009).

Podemos distinguir entre dois tipos de sistemas de recomendação: sistemas *model-based* e sistemas *memory-based*. Esta distinção dá-se através da análise do tipo de abordagem algorítmica utilizada para evidenciar a relevância dos itens(KEIKHOSROKIANI; FYE, 2023). Os sistemas *model-based* se fundamentam em técnicas de machine learning, como por exemplo redes neurais ou factorização matricial, ou seja, utilizam modelos baseados em dados para preverem a relevância dos itens. Por sua vez os *memory-based* fazem uso de fórmulas heurísticas como *nearest-neighbours* para estimarem a similaridade entre os itens ou usuarios(ALMONTE et al., 2022).

A tarefa dos algoritmos baseados em memória diz respeito à previsão da avaliação do usuário a um produto(SHARMA; GERA, 2013).A função de similaridade é uma medida essencial que quantifica a distância entre as avaliações de dois usuários e é utilizada como um peso na predição. Métodos comuns para calcular essa função de similaridade incluem o coeficiente de correlação de Pearson e a similaridade de cosseno. Ambas as medidas dependem da existência de um conjunto suficiente de itens comuns avaliados por ambos os usuários para serem consideradas confiáveis.

2.5 Análise de Sentimentos - Processamento de Linguagem Natural

No contexto atual, marcado pela proeminência de dados digitais, especialmente em plataformas de mídia social, a análise de sentimentos emerge como uma área de pesquisa crucial. Conforme destacado por Nandwani e Verma (2021), esta disciplina busca compreender e categorizar as opiniões e emoções expressas em textos. A análise de sentimentos, situando-se na interseção da ciência da computação, inteligência artificial e linguística, oferece insights significativos sobre as reações e atitudes do público em relação a uma ampla gama de tópicos, desde produtos e serviços até questões políticas e sociais. A análise de sentimento é um meio de avaliar se os dados são positivos, negativos ou neutros.

A participação ativa dos consumidores em avaliar produtos e serviços em plataformas online é crucial para o aprimoramento contínuo de bens e serviços por parte dos fornecedores e prestadores (NANDWANI; VERMA, 2021). Essa interação gera uma rica fonte de dados que, quando analisados, oferecem insights valiosos sobre a percepção do público.

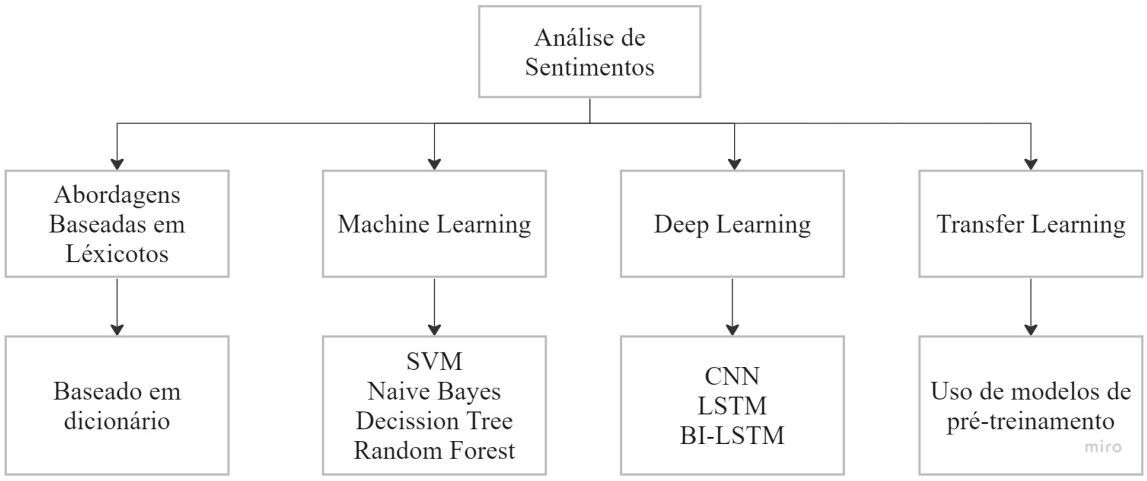
Um dos principais desafios na análise de sentimentos em textos é a necessidade de entender a complexidade da linguagem e como as palavras e frases se relacionam em um contexto mais amplo (LIU et al., 2020). O processo de pré-processamento é fundamental para preparar os dados para análise, envolvendo etapas como tokenização, eliminação de palavras irrelevantes e classificação gramatical das palavras (BHASKAR; SRUTHI; NEDUNGADI, 2015). Métodos como a lematização são empregados para simplificar as palavras para suas formas básicas, reduzindo assim a complexidade do processamento (KRATZWALD et al., 2018), como por exemplo “gostei” e “gostar” se tornam “gosto”.

Além disso, a técnica de *Bag of Words* (BOW) é comumente usada na extração de características textuais. Nesse método, palavras são transformadas em vetores numéricos, facilitando o processamento por algoritmos de aprendizado de máquina (WU; HOI; YU, 2010). Cada palavra é representada por um número baseado em sua presença ou frequência no texto analisado.

A Figura 15 ilustra diversos métodos empregados na análise de sentimentos e na identificação de emoções, divididos principalmente em técnicas lexicais, métodos fundamentados em aprendizado de máquina e estratégias baseadas em aprendizado profundo. Uma técnica híbrida, que mescla métodos estatísticos e de aprendizado de máquina, é utilizada para mitigar as limitações inerentes a cada uma dessas abordagens.

Para avaliar o desempenho de um modelo de análise de sentimentos, podemos gerar

Figura 15 – Técnicas para análise de sentimentos e detecção de emoções.



Fonte: Adaptado de (NANDWANI; VERMA, 2021)

uma matriz de confusão, que fornece a contagem de julgamentos ou previsões corretas e incorretas com base em valores reais conhecidos. Esta matriz exibe valores verdadeiros positivos (TP), falsos negativos (FN), falsos positivos (FP), verdadeiros negativos (TN) para ajuste de dados com base em classes positivas e negativas. Com base nesses valores, os pesquisadores avaliaram seu modelo com métricas como exatidão, precisão e recall, pontuação F1, etc.

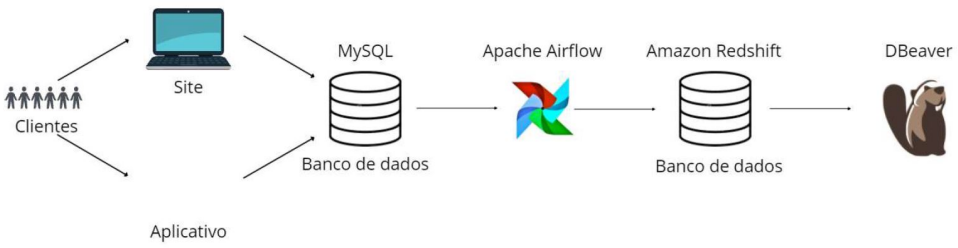
3 METODOLOGIA

Este capítulo é dedicado ao detalhamento da implementação do sistema proposto. As seções estão divididas de forma a viabilizar a reprodução dos resultados obtidos, ressaltando as particularidades encontradas nas etapas de implementação da: (ii) Extração dos dados, (ii) Modelagem dos dados, (iii) Pré-processamento, (iv) Modelagem do Aprendizado de Máquina e (v) Testes e Validações do sistema.

3.1 Extração dos dados

Este projeto visa estudar o ciclo de compras em um e-commerce analisando o comportamento dos clientes, sendo assim, todas as informações utilizadas são provenientes de interações deles com a empresa. Dentro dessas aplicações, os clientes podem realizar procedimentos de busca e compra de produtos, cancelamento de compras, assinatura, cadastro de informações pessoais, indicação de produtos, cadastro e escolha de informações referentes à compras, como endereço de entrega e modo de pagamento, gerenciamento de conta sobre a política de cashback, entre outros.

Figura 16 – Extração dos dados.



Fonte: (LIMA, 2021)

Os dados gerados por diversas operações são armazenados em um banco de dados MySQL, estruturado de forma relacional para otimizar os processos transacionais. Para consultas e análises mais aprofundadas, esses dados são transferidos para o Amazon Redshift, um sistema de banco de dados colunar, por meio do software AirFlow. Esta migração é parte integrante de um processo de ETL (Extração, Transformação e Carga), que é fundamental para a gestão eficiente de dados em um Data Warehouse, sendo crucial para análises e ciência de dados. A extração e consulta desses dados no Amazon Redshift são realizadas utilizando o software DBeaver, conforme ilustrado na Figura 16, empregando a linguagem SQL (Structured Query Language).

Na construção de sistemas de aprendizado de máquina para e-commerce, a seleção e o pré-processamento de dados são etapas cruciais. Inicialmente, a seleção de variáveis relevantes é realizada com base em métodos estatísticos para garantir a relevância dos dados no modelo. Em seguida, o pré-processamento envolve limpeza, normalização, imputação e codificação dos dados, essenciais para manter a integridade e a utilidade do conjunto de dados (MITCHELL, 1997).

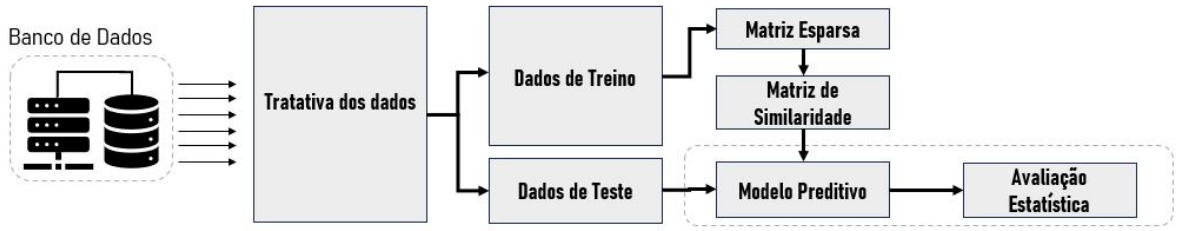
A seleção e o pré-processamento adequados são vitais para extrair insights relevantes, melhorando a experiência de compras online e tornando a modelagem de dados um pilar central na otimização de sistemas de recomendação e análise preditiva. Se quisermos construir um modelo de aprendizado de máquina eficaz, precisamos acertar na extração de características (TAULLI, 2019).

Os modelos propostos, como sistemas de recomendação, previsão de lucros, cancelamento de pedidos e análise de sentimentos, requerem uma modelagem de dados específica para atender às suas necessidades e características únicas.

3.2 Sistema de recomendação

Neste projeto iremos apresentar um sistema de recomendação de filtragem colaborativa baseada na similaridade de itens. O processo envolve as as etapas apresentadas na Figura 17, a etapa que fundamenta a recomendação é, justamente, encontrar a semelhança entre produtos a partir de uma matriz de similaridade e realizar previsão de avaliações a produtos que o cliente ainda não comprou.

Figura 17 – Diagrama da arquitetura do sistema de recomendação.



Fonte: O próprio autor

3.2.1 Seleção e Pré-Processamento dos dados

Neste projeto utilizaremos a filtragem colaborativa que foca no usuário e em outros usuários que são matematicamente similares. Neste contexto, a "Data"oferece insights temporais,

permitindo análises de tendências e variações sazonais nas avaliações dos usuários. O "User"e o "Skucode", representando respectivamente os identificadores do usuário e do produto, são vitais para estabelecer a matriz de interação entre usuários e produtos, que é o núcleo da filtragem colaborativa. Esta matriz captura as avaliações, ou "Ratings", que são essenciais para identificar padrões de preferência e fazer recomendações.

No algorítimo 1, a função `CAST` está sendo aplicado às colunas `user`, `skucode` e `rating`, para que esses campos sejam transformados para identificadores numéricos únicos e pontuações que devem ser representadas por números inteiros. Além disso, a função `DATE` está sendo aplicada à coluna `data_avaliacao` para extrair apenas a componente de data de um valor de data/hora, descartando a parte do tempo se estiver presente. Dessa forma, modelamos os dados da seguinte forma:

Data: consulta SQL para seleção de dados
Result: Dados estruturados para sistema de recomendação

```
seleção de dados de avaliação
begin
    select distinct
            cast(tc.user as int) as "user",
            cast(pd.skucode as int) as "skucode",
            cast(tc.rating as int) as "rating",
            date(tc.data_avaliacao) as "date"
    from tabela_avaliacao tc
    left join tabela_produto pd on pd.id_product = tc.id_product
    where 1=1
        and user is not null
        and skucode is not null
        and rating is not null
        and data_avaliacao is not null
end
```

Algoritmo 1: Consulta SQL para extração e transformação de dados de avaliações

A consulta está selecionando dados de duas tabelas (`tabela_comentario` e `tabela_tproduct`) onde as avaliações dos usuários são armazenadas junto com informações do produto.

Dessa forma, obtemos uma tabela no formato apresentado na Tabela 1:

Os filtros aplicados utilizando os comandos SQL (`WHERE - AND`) visam garantir que nenhum dos campos importantes (`user`, `skucode`, `rating`, `data_avaliacao`) sejam nulo, o que poderia indicar dados incompletos ou inválidos. Ao fazer isso, o código está preparando

Tabela 1 – Atributos do conjunto de dados para Sistema de Recomendação

Variável	Tipo	Descrição	Exemplo
date	date	Data da avaliação	'2023-10-30'
user	float	Código de identificação do usuário	'12345.0'
skucode	float	Código de identificação do produto	'5678.0'
rating	float	Avaliação do usuário ao produto	'4.0'

Fonte: Produção do próprio autor

um conjunto de dados limpo e consistente que poderá ser usado em nosso modelo de recomendação.

3.2.2 Treinamento do modelo

3.2.3 Matriz Esparsa

Como iremos trabalhar com um modelo baseado em filtragem colaborativa e temos os dados pré-processados no formato da Tabela 1, o treinamento passará por criar matriz esparsa, o modelo do Algorítimo 6 pode ser utilizado para treino e teste:

Data: Matriz esparsa para treinamento e teste
Result: Matriz criada e salva em disco

```
1 Criação da matriz esparsa
2 begin
3     # Criamos a matriz esparsa com os dados de treino
4     matriz_esparsa = sparse.csr_matrix((df.rating.values,(df.user.values,
        df.skucode.values)))
5     # Salvamos a matriz esparsa no disco
6     sparse.save_npz("matriz_esparsa.npz", matriz_esparsa)
7     print('Matriz Esparsa de Salva em Disco.')
8 end
```

Algoritmo 2: Código Python para criação e armazenamento da matriz esparsa.

O código do Algorítimo 6 utiliza a biblioteca `scipy`, que oferece boas ferramentas para trabalhar com matrizes esparsas em `Python`. Criamos uma nova matriz esparsa a partir de um dataframe chamado `df`. Este dataframe contém colunas denominadas rating, user, e skucode, que representam as avaliações dos usuários para os produtos. A matriz esparsa é criada utilizando a função `sparse.csr_matrix`, que recebe como argumentos os valores

das avaliações e as posições correspondentes de usuários e produtos para criar a matriz no formato `Compressed Sparse Row` (CSR), um formato eficiente para matrizes esparsas.

3.2.4 Matriz de Similaridade

Na sequência a matriz de similaridade é calculada, definimos uma função `calcula_similaridade_iten` que calcula as similaridades entre produtos com base em suas interações em uma matriz esparsa. A função utiliza a similaridade de cosseno, uma métrica comum em sistemas de recomendação para medir a similaridade entre dois vetores no espaço de características.

Data: Definindo a função Similaridade entre itens
Result: Matriz de similaridade entre itens

```
1 begin
2     def calcula_similaridade_itens(
3                                 sparse_matrix,
4                                 compute_for_few=False,
5                                 top=100,
6                                 verbose=False,
7                                 verb_for_n_rows=20,
8                                 draw_time_taken=True):
9 end
```

Algoritmo 3: Código Python para cálculo da matriz de similaridade entre itens.

Definição da função `calcula_similaridade_itens` tem vários parâmetros para controlar seu comportamento como por exemplo a `sparse_matrix` que é a matriz esparsa de entrada e top que é o número de principais similaridades a serem calculadas. Dentro desta função temos uma etapa de incialização das variáveis e o cálculo de similaridade.

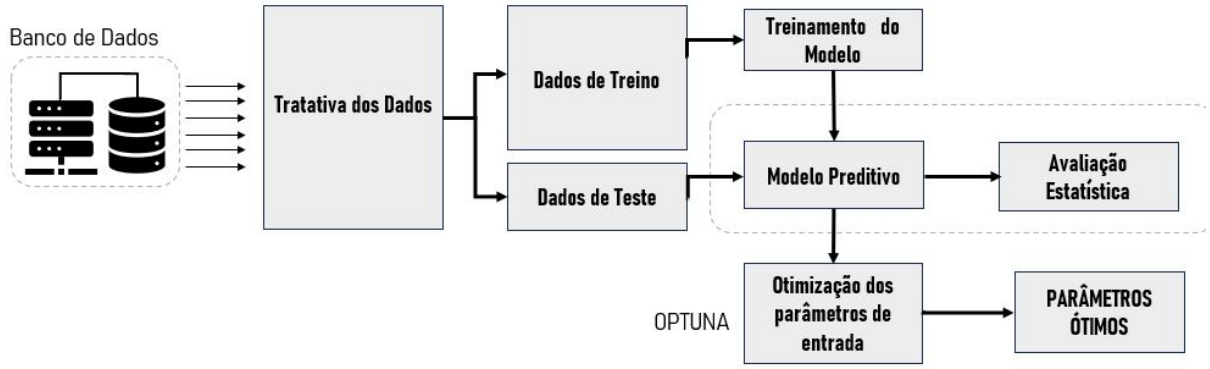
O cálculo de similaridade acontece dentro de um loop, para cada usuário, a função:

- Calcula a similaridade de cosseno, como mostrado no algorítimo 8;
- Armazena os índices dos top 100 usuários mais semelhantes;

3.3 Otimização de Lucro

A otimização do lucro em nosso projeto envolve um processo de duas etapas. Inicialmente, focamos no treinamento de um modelo de redes neurais com o objetivo de provisionar de forma precisa o lucro obtido. Após o treinamento do modelo, a segunda etapa é dedicada à identificação e ajuste dos parâmetros de entrada do modelo que otimizam o lucro previsto. Para a otimização utilizamos o optuna, que automatiza o processo de busca pelos melhores hiperparâmetros para um dado modelo. Neste caso, a função objetivo é maximizar a saída do modelo de rede neural. O diagrama do modelo proposto é apresentada na Figura18.

Figura 18 – Diagrama da arquitetura do modelo de otimização de lucro.



Fonte: O próprio autor

3.3.1 Seleção e Pré-Processamento dos dados

Para o nosso modelo, será escolhido parâmetros de entrada que têm uma influência direta e significativa na previsão do lucro bruto. Parâmetros como 'CMV' (custo da mercadoria vendida) e 'Qtd_itens' oferecem insights cruciais sobre os custos e a demanda, respectivamente. A inclusão da 'Data' reconhece a importância da sazonalidade e das tendências de mercado, como apresentada na consulta SQL no Algorítmo 4.

Data: consulta SQL para seleção de dados

```
1  seleção de dados de vendas
2  begin
3      select distinct
4              date(tp.data_venda) as data_venda,
5              tp.skucode,
6              cmv.cmv_unitario,
7              sum(tp.quantidade) as itens,
8              cast(sum(tp.receita_bruta as decimal(12,2))) as receita_brt,
9              cast(sum(tp.receita_liquida as decimal(12,2))) as receita_lqd,
10             cast(sum(tp.desconto_total as decimal(12,2))) as desconto,
11             lucro_percent**,
12             margem**
13     from tabela_produto tp
14     left join tabela_cmv cmv
15         on (cmv.skucode = tp.skucode
16         and date(cmv.data_cmv) = date(tp.data_venda))
17     where 1=1
18         and tp.skucode = '23266'
19         and date(tp.data_venda) >= date('2019-01-01')
20         and tp.is_kit = 0
21     group by 1, 2, 3
22 2
23 end
```

Algoritmo 4: Consulta SQL para seleção de dados de vendas

As principais tratativas de dados aplicadas nesta etapa consistem em selecionar registros distintos para evitar duplicatas nos resultados, extrair apenas a componente de data de uma coluna e converter as somas de `receita_bruta`, `receita_liquida` e `desconto_total` em valores decimais com duas casas decimais, garantindo a precisão desses indicadores. Além disso, `lucro_percent` e `margem` são determinados por operações matemática e, em seguida, convertido para um formato decimal.

A fórmula utilizada para calcular a percentagem de lucro é expressa como:

$$\text{Lucro Percentual} = \left(\frac{\text{Receita Líquida} \times (1 - \text{Tributos}) - \text{CMV}}{\text{Receita Líquida}} \right) \tag{3.1}$$

Neste cálculo, a (`Receita Líquida`) é o montante total arrecadado após todos os descontos

e devoluções serem aplicados, os (`Tributos`) representam as diversas obrigações fiscais incidentes sobre a receita, e o (`CMV`) é o Custo das Mercadorias Vendidas, ou seja, o custo direto associado à produção dos bens que a empresa comercializa. Esta fórmula resulta na percentagem do lucro líquido em comparação à receita líquida, oferecendo um indicador percentual da eficiência financeira.

Quanto ao lucro líquido, ele é obtido pela fórmula:

$$\text{Lucro Líquido} = \text{Receita Líquida} \times (1 - \text{Tributos}) - \text{CMV} \tag{3.2}$$

Diferente da percentagem de lucro, que oferece uma visão relativa, o lucro líquido oferece uma perspectiva absoluta, indicando o valor monetário do lucro resultante das operações de venda. Ao incorporar essas fórmulas na consulta SQL, calculamos a nossa variável objetivo do sistema de maximização de lucro.

O código SQL apresentado no Algorítimo 2 representa uma versão resumida da seleção de variáveis e, dessa forma, para preparar os dados para um modelo de rede neural que visa provisionar o lucro, ainda é necessário realizar algumas etapas etapas essenciais para a garantir a performance do modelo, como:

1. **Padronização Numérica:** Normalizar ou padronizar as variáveis numéricas para garantir escalas comparáveis. Neste projeto será utilizado a ferramenta `StandardScaler()` fornecida pela biblioteca `scikit-learn` em python.
2. **Conversão de Categorias:** Transformar variáveis categóricas em formatos numéricos usando codificação *one-hot*.
3. **Seleção de Recursos:** Eliminar variáveis com baixa variância ou alta correlação para simplificar o modelo. Pode ser criado uma matriz de correlação utilizado a ferramenta `corr()` fornecida pela biblioteca `pandas` em python.
4. **Divisão de Dados:** Separar os dados em conjuntos de treinamento, validação e teste, utilizando estratificação quando houver desequilíbrio. Podemos utilizar a ferramenta `train_test_split()` que é uma função do módulo `model_selection` da biblioteca `scikit-learn` em Python.

Dessa forma, com o Algorítimo 4 obtemos um conjunto de dados no formato apresentado na Tabela 2:

Tabela 2 – Atributos do conjunto de dados para Regressão

Variável	**Tipo**	**Descrição**	**Exemplo**
date	date	Data em que o produto foi vendido	'2023-10-30'
skucode	float	Código de identificação do produto	'12345.0'
cmv	float	Custo da mercadoria vendida	'12.0'
qtd_itens	float	Quantidade de produtos vendidos	'5678.0'
receita_brt	float	Receita bruta (Antes de aplicação de descontos)	'40,000.0'
receita_lqd	float	Receita bruta (Depois de aplicação de descontos)	'30,000.0'
desconto	float	Desconto total aplicado pelo cliente	'10,000.0'
lucro_percent	float	Margem de lucro percentual	'0.3000'
lucro	float	Lucro Absoluto	'10,000.0'

Fonte: Produção do próprio autor

3.3.2 Treinamento do modelo

3.3.2.1 Modelo de Rede Neural

A primeira etapa do treinamento do modelo de otimização do lucro de vendas passa por criar um modelo de rede neural que seja capaz de provisionar o lucro de um determinado produto. Dessa forma, a implementação do modelo pode ser realizada da seguinte forma:

```
1  Construindo o modelo de rede neural para provisão de lucro
2  begin
3      model = Sequential([
4                          Dense(128, activation='relu', input_dim=X_train.shape[1]),
5                          Dropout(0.2),
6                          Dense(64, activation='relu'),
7                          Dropout(0.2),
8                          Dense(32, activation='relu'),
9                          Dropout(0.2),
10                         Dense(1)
11     ])
12     model.compile(optimizer='adam', loss='mean_squared_error')
13     # Treinando o modelo usando dados de treinamento e validação history =
        model.fit(
14                         X_train,
15                         y_train,
16                         epochs=100,
17                         batch_size=32,
18                         validation_data=(X_val, y_val)
19     )
20 end
```

Algoritmo 5: Código Python da rede neural para provisão de lucro.

3.3.2.2 Otimizando os Hiperparâmetros do Modelo de Rede Neural

Ao desenvolver um modelo de rede neural, a seleção de hiperparâmetros adequados é crucial para garantir a precisão das previsões. A ferramenta Optuna oferece uma solução automatizada para encontrar a combinação ideal de hiperparâmetros. Ela opera explorando diversas combinações e avaliando o desempenho de cada modelo com base em uma métrica de sucesso predefinida, como a perda de validação. Ao focar na minimização desta perda, o Optuna ajusta os hiperparâmetros efetivamente, o que pode melhorar a acurácia das previsões de lucro.

Para implementar a otimização de hiperparâmetros utilizando o Optuna, é necessário definir uma função objetivo. Essa função cria um modelo com um conjunto de hiperparâmetros e retorna o valor a ser minimizado — tipicamente, a perda de validação. O Optuna então testa diferentes combinações de hiperparâmetros para identificar aquela que reduz ao máximo o valor retornado pela função objetivo.

O Optuna é capaz de otimizar uma ampla gama de hiperparâmetros, incluindo, taxa de aprendizado, número de camadas em uma rede neural, número de neurônios em cada camada, parâmetros de regularização como dropout, e parâmetros específicos do modelo, como a profundidade máxima em uma árvore de decisão. Essa abordagem flexível permite uma busca abrangente no espaço de hiperparâmetros para aprimorar o desempenho do modelo.

3.3.2.3 Otimizando os Parâmetros de entrada do Modelo

A otimização dos parâmetros de entrada de um modelo com o objetivo de maximizar o lucro é uma aplicação distinta da otimização de hiperparâmetros. Enquanto a otimização de hiperparâmetros foca na configuração interna do modelo de aprendizado de máquina, a otimização de parâmetros de entrada visa ajustar as variáveis que alimentam o modelo para maximizar o lucro, que é a função objetivo nesse contexto.

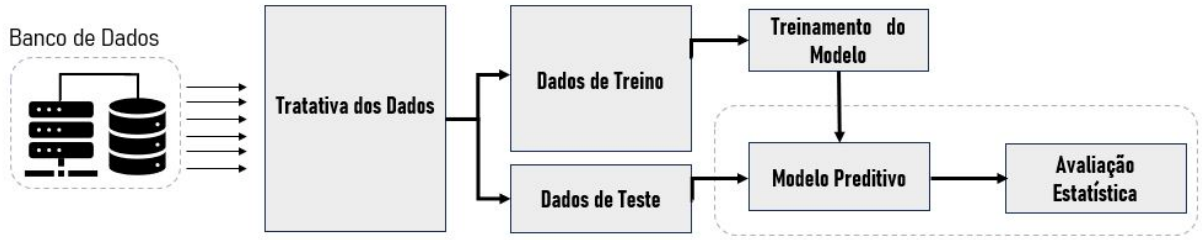
Embora o Optuna não seja projetado especificamente para a otimização de parâmetros de entrada, neste projeto o adaptamos para essa finalidade. Para isso, estabelecemos uma função objetivo que receba parâmetros de entrada e utilize o modelo treinado para retornar o lucro esperado. Diferentemente da otimização de hiperparâmetros, onde o objetivo é minimizar a perda, na otimização de parâmetros de entrada, o Optuna é configurado para maximizar o lucro previsto pelo modelo.

Ao aplicar o Optuna para otimizar tanto hiperparâmetros quanto parâmetros de entrada, é possível não apenas aprimorar a eficiência do modelo de rede neural, mas também identificar as condições ideais que conduzem ao aumento do lucro previsto, proporcionando uma ferramenta valiosa para tomadas de decisão baseadas em dados.

3.4 Provisão de Cancelamento de Pedidos

Para a provisão de cancelamento de pedidos, desenvolvemos um modelo de aprendizado supervisionado baseado em redes neurais, que incorpora etapas fundamentais e específicas, incluindo a preparação e tratativa dos dados, além do treinamento do modelo. O diagrama do modelo proposto é apresentada na Figura19.

Figura 19 – Diagrama da arquitetura do modelo de provisão de cancelamento de pedidos.



Fonte: O próprio autor

3.4.1 Seleção e Pré-Processamento dos dados

A seleção de dados para a provisão de cancelamentos envolve parâmetros da compra, como, por exemplo, os meios de pagamento e o canal de compra, que pode ser via aplicativo ou site web. Também inclui a quantidade de itens comprados, o valor total do pedido e a quantidade de descontos aplicados. Além disso, parâmetros do usuário, como o tempo de cadastro e se é membro ou não de um clube de assinatura, são considerados, bem como os dados históricos do cliente, que verificam a quantidade de cancelamentos anteriores à compra atual e o percentual de cancelamentos. Estes são parâmetros essenciais para aumentar a acurácia do nosso modelo.

Como em todos os modelos deste projeto, realizaremos diversas tratativas de dados por meio de consultas SQL. Essa linguagem possui várias funções que podem ser implementadas de forma estruturada em uma consulta. Podemos criar uma subconsulta trazendo os dados históricos de compras de um usuário, possibilitando, assim, avaliar o histórico a cada nova compra.

A função `LAG()` no SQL podemos acessar dados de uma linha anterior à linha atual dentro de uma partição ou conjunto de resultados de uma consulta. Ela é usada para comparar valores entre linhas que estão em uma sequência específica, como uma série temporal. Será utilizado para as variáveis `acumulado_complet, acumulado_cancel, acumulado_cancel`.

A seguir, será apresentada parte do código SQL que pode ser utilizado para selecionar os dados visando realizar a provisão de cancelamento de pedidos em um e-commerce.

Data: consulta SQL para seleção avançada de dados - Parte 2
Result: Dados estruturados para análise detalhada

```
1  seleção detalhada de dados de compras
2  begin
3      select distinct
4          date_part(year, data_compra) as ano_compra,
5          date_part(month, data_compra) as mes_compra,
6          date_part(day, data_compra) as dia_compra,
7          date_part(year, data_cadastro) as ano_cadastro,
8          date_part(month, data_cadastro) as mes_cadastro,
9          date_part(day, data_cadastro) as dia_cadastro,
10         case
11             when wineup_status = 'estreante' then 1
12             when wineup_status = 'entusiasta' then 2
13             when wineup_status = 'protagonista' then 3
14             when wineup_status = 'estrela' then 4
15             when wineup_status = 'ídolo' then 5
16             else 0
17         end nivel_wineup,
18         datediff(day, data_cadastro, data_compra) as
            tempo_cadastro_compra,
19         case
20             when tipo_compra is not null then 1 else 2
21         end compra_select,
22         case
23             when canal = 'APP' then 1
24             when canal = 'WEB' then 2
25         end canal,
26         case
27             when tipo_cliente = 'Sócio' then 1 else 2
28         end tipo_cliente, - socio = 1 and cliente =2
29         case
30             when pagamento = 'APPLE_PAY' then '1'
31             when pagamento = 'CREDITCARD' then '2'
32             when pagamento = 'GOOGLE_PAY' then '3'
33             else 9
34         end pagamento,
35 end
```

Algoritmo 6: Consulta SQL avançada para seleção de dados de compras - Parte 1

Data: consulta SQL para seleção avançada de dados - Parte 2

```
1  begin
2          case
3              when ord = 1 then 0.001 else rec
4          end recencia,
5          case
6              when desembolsado is null then 0 else desembolsado
7          end desembolsado,
8          case
9              when desconto_cupom is null then 0 else desconto_cupom
10         end desconto_cupom,
11         case
12             when desconto_cashback is null then 0 else desconto_cashback
13         end desconto_cashback,
14         qtd_itens,
15         ord recorrencia,
16         Acumulado_compl,
17         Acumulado_cancel,
18         percentual_cancelam,
19         case
20             when status = 'CANCELLED' then 1 else 0
21         end target,
22         row_number() over (partition by target) aux
23     from tabela_pedidos tp
24     left join tabela_cadastro cd
25             on cd.user_id = tp.user_id
26     left join tabela_vendas_assistidas va
27             on va.numero_pedido = tp.numero_pedido
28     left join tabela_socios sc
29             on sc.user_id = tp.user_id
30     left join tabela_pagamento pg
31             on pg.numero_pedido = tp.numero_pedido
32     left join tabela_historico_compras hc
33             on hc.numero_pedido = tp.numero_pedido
34     where 1=1
35         and date(data_compra) >= date(20190101)
36         and status in ('COMPLETED', 'CANCELLED')
37 end
```

Algoritmo 7: Consulta SQL avançada para seleção de dados de compras - Parte 2

Dessa forma, obtemos os dados da seguinte forma:

Tabela 3 – Atributos do conjunto de dados para Classificação

Variável	**Tipo**	**Descrição**	**Exemplo**
dia_compra	float	Dia da compra	‘30’
mes_compra	float	Mês da compra	‘10’
ano_compra	float	Ano da compra	‘2023’
dia_cadastro	float	Dia de cadastro do cliente	‘29’
mes_cadastro	float	Mês de cadastro do cliente	‘10’
ano_cadastro	float	Ano de cadastro do cliente	‘2023’
tempo_cad_comp	float	Dias entre Cadastro e compra	‘1’,‘20, ‘50’
status_cliente	float	Status do Cliente	‘1’, ‘5’, ‘15’
compras_select	float	Qauantidade compras assistidas	‘1’, ‘2’, ... ‘5’
canal	float	Canal de realização da compra	‘1’,‘2’
tipo_cliente	float	Menbro do clube de assinaturas	‘1’,‘2’
pagamento	float	Meio de pagamento	‘1’,‘2’,‘3’
recencia	float	tempo em dias da última compra	‘10,‘37’,‘132’
desembolsado	float	Valor que o cliente pagou	‘220.0’
desc_cupom	float	Valor descontado em cupom	‘10’
desc_cashback	float	Valor descontado em cashback	‘5’
qtd_itens	float	Quantidade de itens no pedido	‘5’
recorrencia	float	Quantidade de pedidos do cliente	‘15.0’
h_compl	float	Quantidade de pedidos do completos	‘10.0’
h_cancel	float	Quantidade de pedidos do cancelados	‘5.0’
hp_cancelam	float	Percentual de cancelamento	‘0.3333’
target	float	Status do pedido	‘1’

Fonte: Produção do próprio autor

Apesar de termos todos os dados em formato numérico, ainda será necessário realizar algumas etapas adicionais para assegurar a performance do modelo de modelo de rede neural destinado à provisão de cancelamento de pedido como:

1. **Padronização Numérica:** Normalizar ou padronizar as variáveis numéricas para garantir escalas comparáveis. Neste projeto será utilizado a ferramenta `StandardScaler()` fornecida pela biblioteca `scikit-learn` em python.

2. **Seleção de Recursos:** Eliminar variáveis com baixa variância ou alta correlação para simplificar o modelo. Pode ser criado uma matriz de correlação utilizado a ferramenta `corr()` fornecida pela biblioteca `pandas` em python.

3. **Divisão de Dados:** Separar os dados em conjuntos de treinamento, validação e teste, utilizando estratificação quando houver desequilíbrio. Podemos utilizar a ferramenta `train_test_split()` que é uma função do módulo `model_selection` da biblioteca `scikit-learn` em Python.

3.4.2 Treinamento do modelo

Neste modelo é utilizado a classe `MLPClassifier` da biblioteca `scikit-learn` para implementar um modelo de Perceptron Multicamadas (MLP), que é um tipo de rede neural feedforward utilizada em problemas de classificação.

- `hidden_layer_sizes=(50, 30)`: Define a arquitetura das camadas ocultas do MLP. No modelo especificado, existem duas camadas ocultas com 50 e 30 neurônios, respectivamente.
- `max_iter=100`: Especifica o número máximo de iterações de treinamento sobre o conjunto de dados. O treinamento é interrompido após 100 iterações, independentemente da convergência do modelo.

O método `partial_fit` é utilizado para treinar o modelo incrementalmente:

```
model.partial_fit(X_treino, y_treino, classes=np.unique(y_treino))
```

Onde:

- `X_treino`: Dados de entrada para treinamento.
- `y_treino`: Rótulos correspondentes às entradas.
- `classes=np.unique(y_treino)`: Lista de todas as classes possíveis no conjunto de dados.

O treinamento do modelo MLP é supervisionado, o que implica que o modelo aprende a mapear as entradas para as saídas desejadas com base em exemplos fornecidos durante o treinamento.

Neste modelo também é possível aplicar a ferramenta `optuna` para encontrar otimizar os hiperparâmetros de entrada do modelo.

3.5 Análise de Sentimentos

3.5.1 Seleção e Pré-Processamento dos dados

Para a análise de sentimentos cada comentário é uma expressão direta da opinião do cliente, serve como uma janela para entender suas experiências e sentimentos. Estes comentários são então categorizados com rótulos de "1"quando positivo ou "0"quando negativo. Estes rótulos foram atribuídos com base no teor do feedback do cliente, oferecendo uma visão binária da satisfação do cliente.

Data: Consulta SQL para rotulagem de dados
Result: Rótulos estruturados para análise de texto

```
1  Pré-processamento e classificação de comentários
2  begin
3      select distinct
4              trim(translate(lower("text"),
5              'áàâãäéèêëíìïóòôõöúùûüçáàâãäéèêëíìïóòôõöúùûüç',
               'aaaaaeeeeiiiiooooouuuucaaaaaeeeeiiiiooooouuuuc')))
6      as texto,
7      begin
8          case
9              when texto like '%ruim%' then 0
10             when texto like '%horror%' then 0
11             when texto like '%pessimo%' then 0
12             when texto like '%nao gostei%' then 0
13             when texto like '%aquem%' then 0
14             ...// mais condições negativas
15             when texto like '%ador%' then 1
16             when texto like '%gost%' then 1
17             when texto like '%bom%' then 1
18             when texto like '%espetacul%' then 1
19             when texto like '%delici%' then 1
20             ...// mais condições positivas
21         else null
22     end
23     as rotulo
24     from tabela_comentarios
25 end
```

Algoritmo 8: Consulta SQL para rotulagem de dados com base no conteúdo de texto

Na consulta SQL detalhada no Algoritmo 8, a função `LIKE` é empregada para rotular comentários, executando buscas por padrões específicos em strings de texto. Utilizando o símbolo `%` como um coringa, é possível criar padrões de pesquisa flexíveis. Por exemplo, a expressão `texto LIKE 'a%'` recuperaria todos os registros onde o campo de texto inicia com a letra 'a'. Da mesma forma, `texto LIKE '%adorei%'` selecionaria registros que contêm a palavra "adorei"em qualquer posição do campo de texto.

Tabela 4 – Atributos do conjunto de dados para Análise de Sentimentos

Variável	**Tipo**	**Descrição**	**Exemplo**
texto	string	Comentário do cliente	'Este produto é excelente!!'
rotulo	string	Rótulo do comentário	'Positivo'

Fonte: Produção do próprio autor

Para este projeto iremos abordar de forma simplificada por meio de dois parâmetros sendo texto e rótulo, dessa forma, podemos implementar de forma rápida e fornecer uma base sólida para análises mais profundas. Para realizar uma análise de sentimentos são realizadas algumas etapas de pré-processamento de texto:

1. **Tokenização:**Separa o texto em palavras ou tokens. Isso geralmente envolve a remoção de pontuação e a divisão com base em espaços e outros delimitadores.
2. **Normalização:**Converte todo o texto para minúsculas para garantir que palavras como “Bom” e “bom” sejam tratadas como a mesma palavra.
3. **Remoção de Stop Words:** Exclue palavras comuns (por exemplo, “e”, “o”, “a”, “com”) que não contêm informações significativas para a análise de sentimentos.
4. **Stemming e Lemmatização:** Reduz as palavras às suas formas raiz (stemming) ou suas formas lexicais básicas (lemmatização).
5. **Tratamento de Negativas:** Trata negativas de forma adequada. Por exemplo, a frase “não é bom” tem um significado diferente de “é bom” e deve ser tratada de maneira diferente na análise.
6. **Vetorização:** Converte o texto limpo em um formato numérico (vetores) que pode ser usado por algoritmos de *Machine Learning*. Técnicas comuns incluem *Bag of Words*, e *Word Embeddings*.
7. **Equilíbrio de Classes:** Garante que o conjunto de dados tenha uma distribuição balanceada de diferentes sentimentos para evitar viés no modelo.

3.5.2 Treinamento do modelo

Data: Implementando um processo de redução de dimensionalidade
Result: Matriz de dimensionalidade reduzida por SVD

```
1  Iimplementando um processo de redução de dimensionalidade
2  begin
3      # Construindo o Modelo
4      model = Sequential([
5                          model = Sequential()
6                          model.add(Embedding(vocab_size, embed_dim,
                            input_length=max_length))
7                          model.add(LSTM(64, dropout=0.2, recurrent_dropout=0.2))
8                          model.add(Dense(2, activation='softmax'))
9      ])
10     model.compile(optimizer='adam', loss='categorical_crossentropy',
        metrics=['accuracy'])
11     # Treinando o modelo usando dados de treinamento e validação model.fit(
12                         X_train_pad,
13                         y_train_cat,
14                         batch_size=32,
15                         epochs=10,
16                         validation_data=(X_test_pad, y_test_cat),
17     verbose=2 )
18 end
```

Algoritmo 9: Código Python da redução .

A primeira camada do modelo é a camada de `embedding`, responsável por transformar as palavras representadas por índices inteiros em vetores densos de um tamanho fixo. Ela permite que o modelo aprenda o significado de cada palavra e as relações semânticas entre elas durante o treinamento. O tamanho do vocabulário é definido com base no número de palavras únicas no conjunto de treino, e a dimensão do `embedding` é um hiperparâmetro que estabelece o tamanho do vetor de saída da camada de `embedding`.

Após a camada de `embedding`, uma camada LSTM é adicionada. Essa camada é adequada para aprender dependências de longo prazo no texto. A LSTM é menos propensa ao problema do esquecimento de informações importantes, comum em RNNs tradicionais. Introduzimos parâmetros de regularização como `dropout` e `recurrent dropout` para evitar o sobreajuste.

A última camada é uma camada densa com ativação `softmax` que converte as saídas da LSTM em probabilidades para cada classe de sentimento. O número de neurônios nesta camada corresponde ao número de classes de sentimentos a serem previstos.

O modelo é compilado com o otimizador `adam` e a função de perda *categorical crossentropy*. A acurácia é utilizada como métrica para monitorar o desempenho durante o treinamento.

O modelo é treinado com os dados de treino tokenizados e padronizados, utilizando o método `fit`. Definimos um tamanho de `batch` e um número de `epochs` para o treinamento. Um conjunto de validação é utilizado para monitorar o desempenho do modelo em dados não vistos durante o treinamento. O treinamento fornece atualizações sobre a perda e acurácia após cada época.

Como resultado, obtemos um modelo capaz de distinguir entre sentimentos positivos e negativos em textos. O modelo pode então ser utilizado para fazer previsões sobre novos dados e avaliar sua capacidade de generalização.

3.5.3 Avaliação do Modelo

A validação de um modelo de rede neural, seja em um problema de classificação, como a análise de sentimentos, ou em um problema de regressão, envolve várias etapas e métodos para garantir que o modelo é eficaz e generaliza bem para dados não vistos. Após o treinamento, a avaliação do modelo é feita usando um conjunto de dados de teste independente.

Para os problemas de classificação:

		Classe Predita		Total
		Positiva	Negativa	
Classe Real	Positiva	VP	FN	VP+FN
	Negativa	FP	VN	FP+VN
Total Predito		VP+FP	FN+VN	VP+VN+FP+FN

Tabela 5 – Matriz de confusão com variáveis

- **Acurácia:** A porcentagem de previsões corretas do total de previsões feitas, calculada por:

$$\text{Acurácia} = \frac{\text{Número de previsões corretas}}{\text{Total de previsões}} \tag{3.3}$$

- **Precisão:** A proporção de previsões positivas corretas em relação ao total de previsões positivas feitas, dada por:

$$\text{Precisão} = \frac{\text{Verdadeiros Positivos (VP)}}{\text{VP} + \text{Falsos Positivos (FP)}} \tag{3.4}$$

- **Recall (Sensibilidade):** A proporção de positivos reais que foram identificados corretamente, calculada como:

$$\text{Recall} = \frac{\text{VP}}{\text{VP} + \text{Falsos Negativos (FN)}} \tag{3.5}$$

- **F1-Score:** A média harmônica de Precisão e Recall, definida como:

$$F1 = 2 \times \frac{\text{Precisão} \times \text{Recall}}{\text{Precisão} + \text{Recall}} \tag{3.6}$$

Para os problemas de Regressão:

- **Erro Quadrático Médio (MSE):** A média dos quadrados dos erros, definida como:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.7}$$

  onde y_i são os valores reais e $\hat{y}_i$ são os valores previstos.

- **Erro Absoluto Médio (MAE):** A média do valor absoluto dos erros, dada por:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3.8}$$

- **Coeficiente de Determinação (R^2):** Representa a proporção da variância para a variável dependente que é explicada pelas variáveis independentes no modelo, calculado por:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{3.9}$$

  onde $\bar{y}$ é a média dos valores reais.

No próximo capítulo serão apresentado os problemas, solução proposta e os respectivos resultados obtidos, assim como a avaliação dos modelos obtidos.

4 SIMULAÇÕES E RESULTADOS

Neste capítulo serão apresentados os resultados obtidos nos sistemas propostos sendo eles o sistema de recomendação, um regressor e otimizador de lucro, um classificador de cancelamento de pedidos e um sistema de análise de sentimentos. De tal modo, o capítulo inicia descrevendo o banco de dados utilizado para treinamento e teste das abordagens propostas, também são apresentados todos os resultados obtidos em diferentes etapas do processo, mostrando a evolução obtida a partir da implementação de algumas técnicas apresentadas anteriormente. Este capítulo visa fornecer uma compreensão abrangente das etapas de implementação e avaliação, destacando as contribuições e eficácia do sistema proposto.

4.1 Recursos Computacionais

Para todos os experimentos realizados neste trabalho, foi utilizada a linguagem Python (ROSSUM, 1995) para construção dos modelos de aprendizado de máquina. O software utilizado para escrita do código foi a plataforma Jupyter Notebook Studio, pois permite a combinação de código, visualizações e narrativas textuais. Além disso, foram utilizadas as bibliotecas Keras e Scikit-Learn (PEDREGOSA, 2011), para a etapa de pré-processamento de dados foram usadas as bibliotecas Numpy (OLIPHANT, 2016), Matplotlib (HUNTER, 2007) e Pandas (MCKINNEY, 2010).

Além disso, a máquina utilizada nos experimentos possuía a seguinte configuração: (i) sistema operacional Windows 10; (ii) processador Intel64 Family 6 Model 140 Stepping 1, GenuineIntel; (iii) memória RAM de 16 GB; (iv) unidade de armazenamento de 1TB (disco rígido); (v) placa de vídeo NVIDIA GeForce MX450 1080, com 8 GB de memória dedicada.

4.2 Simulações do Sistema de Recomendação

4.2.1 Distribuição dos Dados

Em nosso conjunto de dados apresentamos a relação entre usuários e produtos, evidenciado por 924.949 avaliações de 90.649 usuários distintos à 9.536 produtos, como mostrado na Tabela 6.

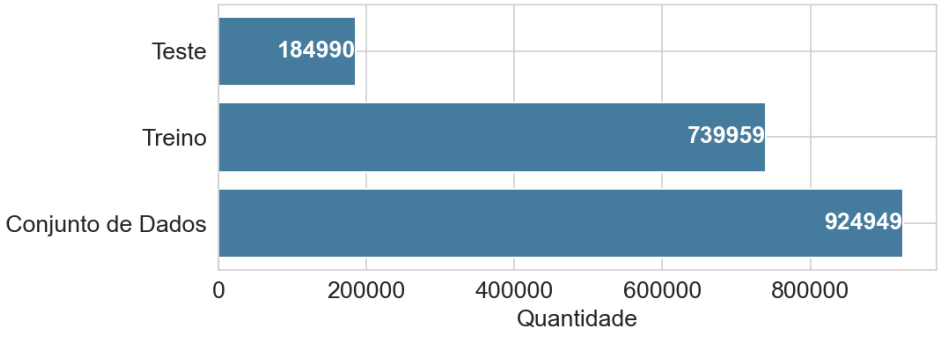
Tabela 6 – Resumo dos dados para a Recomendação

Atributo	**Quantidade**
Número Total de produtos:	9.536
Número Total de Usuários:	90.649
Número Total de Avaliações:	924.949

Fonte: Produção do próprio autor

Vamos dividir os dados em treino e teste utilizando a proporção 80 e 20 para treino e teste, respectivamente, como apresentado na Figura 20.

Figura 20 – Distribuição dos dados para Regressão.



Fonte: O próprio autor

Ao analisar os dados de treino, observamos que 8.607 produtos foram avaliados 739.959 vezes por 76.767 usuários únicos, como apresentado na Tabela 7, o que ressalta uma boa variedade de produtos e uma base sólida para treinar nossos algoritmos.

Por outro lado, os dados de teste, com 4.143 produtos avaliados 184.990 vezes por 23.575 usuários, como apresentado na Tabela 8, permitindo uma validação robusta e confiável dos modelos de recomendação.

Tabela 7 – Resumo dos dados de Treino para a Recomendação

Atributo	**Quantidade**
Número Total de produtos:	8.607
Número Total de Usuários:	76.767
Número Total de Avaliações:	739.959

Fonte: Produção do próprio autor

Tabela 8 – Resumo dos dados de Teste para a Recomendação

Atributo	**Quantidade**
Número Total de produtos:	4.143
Número Total de Usuários:	23.575
Número Total de Avaliações:	184.990

Fonte: Produção do próprio autor

Essa estrutura de dados detalhada e bem distribuída é o coração de um sistema de recomendação eficaz, possibilitando previsões personalizadas e melhorando a experiência do usuário em plataformas de e-commerce.

4.2.2 O Desafio do Cold Start em Sistemas de Recomendação

Em nosso projeto, observamos que uma proporção considerável dos usuários e produtos não faz parte dos dados de treinamento. Especificamente, encontramos que 16.29% dos usuários e 9.99% dos produtos não estão incluídos nos dados de treino. Esta situação representa um desafio significativo, pois sem dados históricos sobre esses usuários ou produtos, o sistema de recomendação enfrenta dificuldades em gerar recomendações personalizadas e precisas.

4.2.3 Criando a Matriz Esparsa

Uma matriz esparsa com uma esparsidade de aproximadamente 100% é extremamente vazia, o que significa que quase todas as suas entradas são zero. Isso é comum em sistemas de recomendação, onde o número de interações (como avaliações de produtos) é muito pequeno em comparação com o número total de possíveis interações entre usuários e itens.

Assim, ao criar uma matriz esparsa do tipo CSR (Compressed Sparse Row) usando a biblioteca scipy.sparse em Python, representamos um sistema de avaliação de item-usuário,

onde cada elemento (m, n) da matriz contém a avaliação do item `m` para o usuario `n`, e a maioria dos elementos da matriz é nulo ou zero, obtemos uma matriz como mostrado na Figura 21.

Figura 21 – Formato da Matriz Esparsa criada.

	skucode_1	skucode_2	skucode_3	...	skucode_m
user_1	rating_1,1	rating_1,2	-		rating_1,m
user_2	-	rating_2,2	rating_2,3		rating_2,m
user_3	rating_3,1	-	rating_3,3		rating_3,m
⋮					
user_n	rating_n,1	rating_n,2	rating_n,3		rating_n,m

Fonte: O próprio autor

A Criação da matriz esparsa de teste segue o mesmo padrão de codificação e formato matricial.

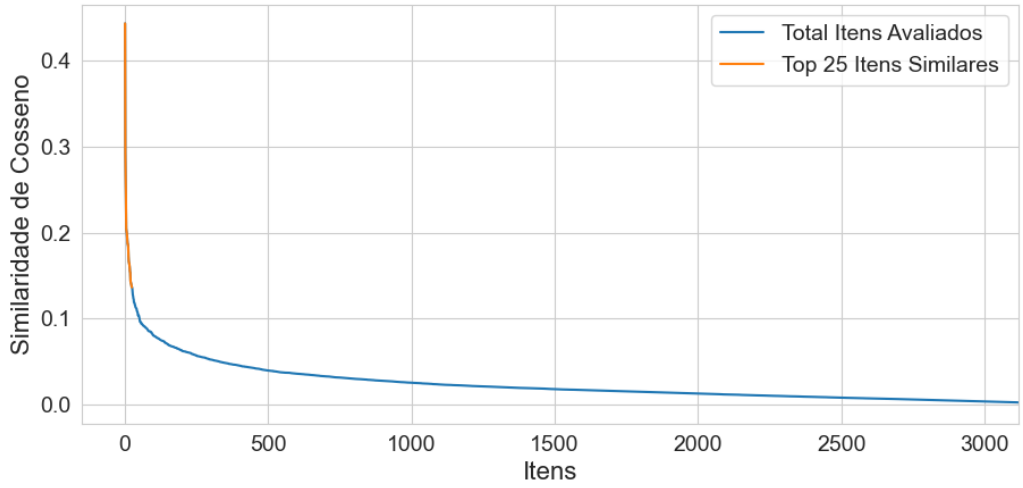
4.2.4 Cálculo de Similaridade

A similaridade de cosseno mede o cosseno do ângulo entre dois vetores no espaço n-dimensional. O valor varia de -1 a 1, onde 1 significa que os vetores são idênticos, 0 indica ortogonalidade (sem semelhança) e -1 significa que os vetores são opostos.

Em Sistemas de Recomendação a similaridade cosseno é utilizada para medir a similaridade entre diferentes usuários ou itens e neste projeto utilizamos a similaridade de itens. Na Figura 22 conseguimos observar o gráfico de similaridade por quantidade de itens quando analisamos o produto de `id = '25270'`, dessa forma conseguimos selecionar os top 25 itens similares baseado na interação de avaliação de usuários aos ítens de treinamento.

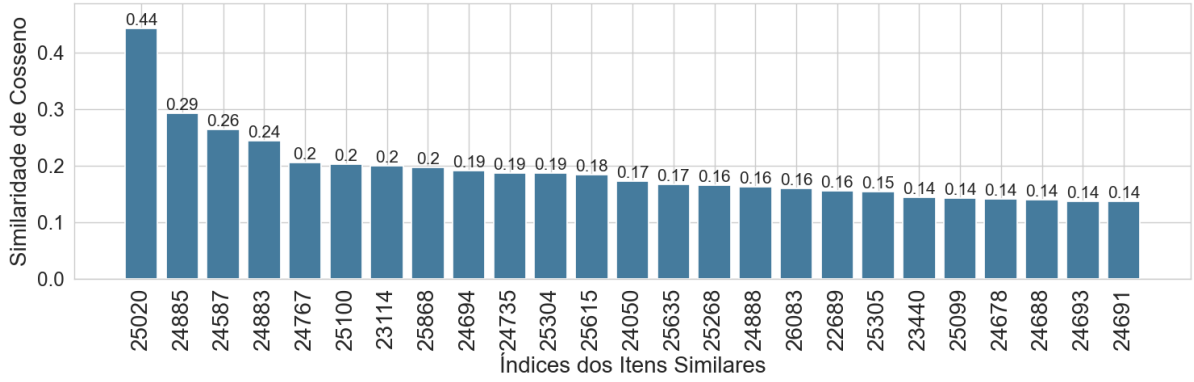
Na Figura 23 listamos em um histograma os 25 itens com maior índicie de similaridade com o produto de referência. Neste caso, os clientes que avaliaram positivamente o produto de `id = '25270'` receberiam uma recomendação do produto de `id = '25020'`, visto que é o produto com maior similaridade (0,44).

Figura 22 – Valores de Similaridade de Cosseno para Itens Similares ao item ’25270’ .



Fonte: O próprio autor

Figura 23 – Top 25 itens similares Similares ao item ’25270’ .



Fonte: O próprio autor

4.2.5 Avaliação do sistema de recomendação

Os resultados apresentados na Tabela 9 nos mostra que que o modelo tem um desempenho global moderado, uma acurácia de 0,6617 significa que o modelo faz previsões corretas em aproximadamente 66,17% das vezes, o que se apresenta como uma eficácia razoável em suas recomendações. O F1 Score é uma medida que combina precisão e recall, em que a precisão é a proporção de classificações corretas em relação ao total de itens classificados e o recall é a sensibilidade, mede a proporção de classificações corretas em relação ao total de classificações reais. Um F1 Score de 0.6593 indica um equilíbrio moderado entre precisão e recall no modelo. Essas métricas sugerem que o modelo é relativamente preciso, porém, há espaço para melhorias.

Tabela 9 – Avaliação das Métricas de Desempenho do Modelo de Recomendação

Métrica	**Valor**
Acurácia	0,6617
F1 Score	0,6593

Para aprimorar o desempenho do modelo de sistema de recomendação baseado em filtragem colaborativa, várias estratégias podem ser adotadas. Uma delas é a implementação de sistemas híbridos, que combinam filtragem colaborativa com métodos como filtragem baseada em conteúdo. Isso não só melhora a precisão das recomendações, mas também ajuda a resolver o problema de cold start enfrentado por novos usuários ou itens com poucas interações.

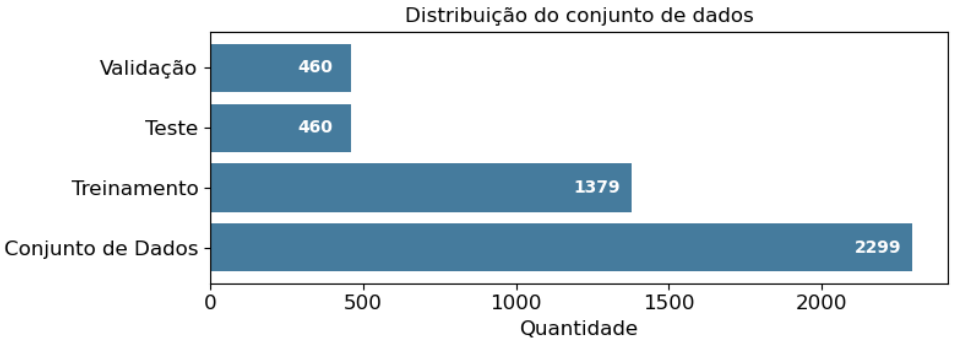
Outra abordagem é a otimização dos parâmetros do modelo, o que pode incluir ajustes no número de vizinhos na similaridade de itens ou na definição do limiar para a classificação binária. Além disso, técnicas de redução de dimensionalidade, como a Decomposição em Valores Singulares (SVD), podem ser eficazes para conjuntos de dados grandes e esparsos.

4.3 Simulações da Otimização de Lucro

4.3.1 Distribuição dos Dados

A distribuição escolhida é uma boa prática padrão. O dataset total, contendo 2.299 observações, foi dividido seguindo a estratégia de validação comum em aprendizado de máquina, visando garantir que o modelo possa ser treinado eficazmente e testado contra dados inéditos.

Figura 24 – Distribuição dos dados para Regressão.



Fonte: O próprio autor

4.3.2 Avaliação do modelo de regressão

O resultado de MSE (Mean Square Error) de 57.386 na avaliação do modelo de regressão, conforme apresentado na Tabela 10, indica que as previsões do modelo desviam, em média, cerca de 57 unidades dos valores reais. A adequação deste valor de RMSE depende significativamente da escala dos dados em questão. Por exemplo, se os valores reais variarem predominantemente entre 100 e 1000, então um RMSE de 57 seria considerado bastante preciso. No entanto, se os valores reais oscilarem entre 0 e 100, então um RMSE de 57 representaria um desvio proporcionalmente alto, sugerindo uma precisão menor do modelo.

Tabela 10 – Avaliação do modelo de Regressão

Métrica	**Valor**
MSE	57,38
MAE	27,96
R^2	0,98

O MAE é a média do valor absoluto dos erros fornece a magnitude do erro, mas, ao contrário do MSE, não penaliza tanto os outliers. Um MAE de 27,92 sugere que, em média, o modelo erra por cerca de 28 unidades. Assim como o RMSE, se isso é alto ou baixo depende da escala dos dados reais

$\mathbf{R^2}$ (Coeficiente de Determinação) é uma medida de quão bem as previsões do modelo se aproximam dos valores reais. Um $\mathbf{R^2}$ de 0.9896 é muito próximo de 1, o que sugere que o modelo explica aproximadamente 98,96% da variação dos dados. Isso geralmente indicaria um ajuste excelente do modelo. Dessa forma, as métricas apontam que o modelo tem um ajuste muito bom aos dados, com um $\mathbf{R^2}$ particularmente alto indicando um alto nível de explicação da variação dos dados.

4.3.3 Otimização dos parâmetros de entrada - Optuna

Tradicionalmente, técnicas de otimização de hiperparâmetros são aplicadas para aprimorar o desempenho dos modelos de aprendizado de máquina. No entanto, neste trabalho, adotou-se uma abordagem inovadora: o Optuna, um framework de otimização de hiperparâmetros, foi utilizado não para aprimorar o modelo, mas para identificar os valores ótimos das variáveis de entrada. Sua utilização tem por finalidade maximizar a saída do modelo, neste caso, a margem de lucro do produto, considerando variáveis como custos, preço de venda e demanda de mercado.

Essa técnica de otimização de parâmetros de entrada, ou otimização de produto, envolve a simulação de diferentes cenários e a identificação daquele que resulta na maior margem de lucro possível. O processo é automatizado e iterativo, permitindo a rápida convergência para a configuração ideal. O resultado da otimização é apresentado na Tabela 11.

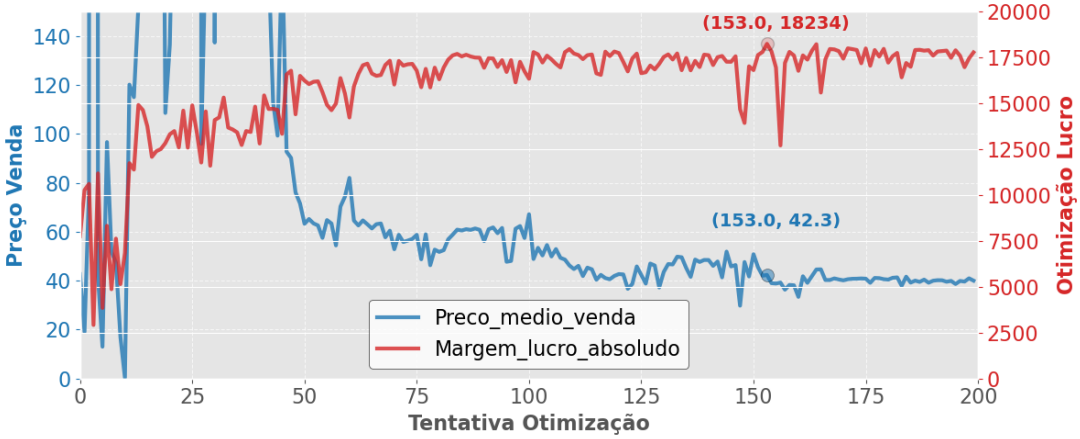
Tabela 11 – Descrição dos Hiperparâmetros do Modelo Neural

Parâmetro	**Descrição**	**Valor Otimizado**
Preço Venda Unitário	Preço de venda do produto	R$ 42.30
Provisão Lucro	Lucro provisionado ótimo	R$ 18.233,00
CMV	Custo de Mercadoria Vendida	R$ 14.556,00
itens	Quantidade de itens vendidos	1.500
receita Bruta	Receita total de venda do produto	R$ 63.516,00
receita liquida	Receita total com descontos	R$ 54.057,00
descontos	Descontos totais	R$ 9.459,00

Fonte: Produção do próprio autor

Após a otimização, os parâmetros que resultaram na maior margem de lucro foram utilizados para determinar o preço médio de venda do produto, como mostrado na Figura 25. Esse preço representa o ponto ideal onde a empresa maximiza seu lucro e mantém a competitividade no mercado. A estratégia vai além de uma aplicação técnica, impactando diretamente na tomada de decisão estratégica e na rentabilidade empresarial.

Figura 25 – Distribuição dos dados para classificação.



Fonte: O próprio autor

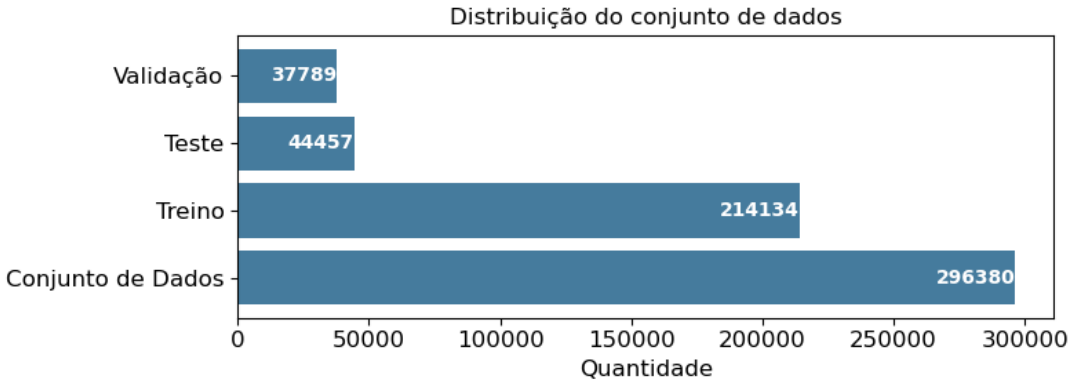
A implementação dessa metodologia representa uma contribuição relevante para o campo da otimização de negócios através de inteligência artificial. A capacidade de ajustar continuamente os parâmetros de entrada em resposta a mudanças no mercado evidencia a flexibilidade e o potencial do modelo para apoiar decisões de negócios dinâmicas e baseadas em dados.

4.4 Simulações da Provisão de Cancelamento de Pedidos

4.4.1 Distribuição dos Dados

Para treinar e validar o modelo de classificação de cancelamento de pedidos, utilizou-se um conjunto de dados dividido em treinamento e teste e validação. A Figura 26 mostra a distribuição dos dados totais, os dados de treinamento e os dados de teste e validação.

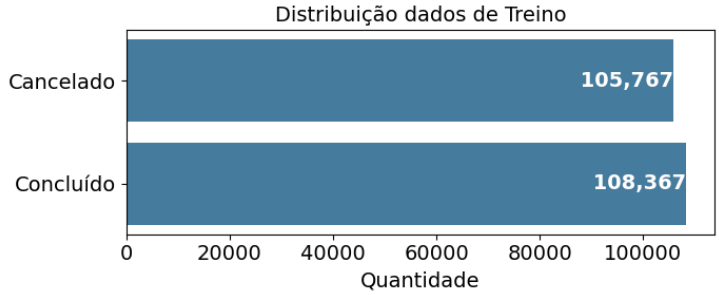
Figura 26 – Distribuição dos dados para classificação.



Fonte: O próprio autor

Como podemos verificar na Figura 27, os dados de teste estão devidamente balanceados, isso é importante, principalmente, para que o nosso modelo não tenha uma tendência de aprendizado e, consequentemente, na sua predição.

Figura 27 – Distribuição dos dados para classificação.



Fonte: O próprio autor

4.4.2 Hiperparâmetros do Modelo

O modelo foi construído utilizando uma arquitetura de rede neural multicamadas (MLP). A Tabela 16 descreve os hiperparâmetros utilizados para o modelo.

Tabela 12 – Descrição dos Hiperparâmetros do Modelo

Hiperparâmetro	**Descrição**	**Valor**
activation	Função de Ativação	relu
alpha	Termo de Regularização L2	0,0001
batch_size	Tamanho do Lote	auto
early_stopping	Parar Antecipadamente	False
epsilon	Valor para Estabilidade Numérica em Adam	1e-08
hidden_layer_sizes	Tamanhos das Camadas Ocultas	(50, 30)
learning_rate	Taxa de Aprendizado	constant
learning_rate_init	Inicialização da Taxa de Aprendizado	0,001
max_iter	Número Máximo de Iterações	100
momentum	Momentum	0,9
n_iter_no_change	Número de Iterações sem Melhoria para Parar	10
solver	Solucionador	adam
tol	Tolerância para Otimização	0,0001
validation_fraction	Fração de Dados para Validação	0,1

Fonte: Produção do próprio autor

Na sequência utlizamos o Optuna que é uma biblioteca de otimização de hiperparâmetros. A função *objective* é definida para receber um argumento *trial*, que representa uma tentativa de otimização de hiperparâmetros.

A função `objective` é utilizada para definir o espaço de busca dos hiperparâmetros para o `MLPClassifier`. Os hiperparâmetros são selecionados da seguinte forma:

- **hidden_layer_sizes**: Número de neurônios na camada oculta. Selecionado de forma logarítmica entre 10 e 150.
- **learning_rate_init**: Taxa inicial de aprendizado. Escolhida de forma logarítmica entre 1×10^{-4} e 1×10^{-1}.
- **alpha**: Parâmetro de penalidade L2 (termo de regularização). Escolhido de forma logarítmica entre 1×10^{-5} e 1×10^{-1}.
- **activation**: Função de ativação para as camadas ocultas. Pode ser `"relu"`, `"tanh"`, ou `"logistic"`.

Avaliamos o modelo fazendo previsões no conjunto de validação `(X_validacao)`, calcula a acurácia comparando as previsões com os rótulos verdadeiros `(y_validacao)` e retorna

a acurácia negativa. O uso da acurácia negativa é comum em contextos de otimização onde se deseja maximizar uma métrica (a acurácia, neste caso), pois muitas bibliotecas de otimização minimizam o valor de retorno da função objetivo por padrão.

As otimizações realizadas nos hiperparâmetros do modelo resultaram nas seguintes alterações, como detalhado na Tabela 13:

Tabela 13 – Alterações nos Hiperparâmetros do Modelo

Hiperparâmetro	**De**	**Para**
hidden_layer_sizes	(50, 30)	124
learning_rate_init	0,001	0,0051
alpha	0,0001	$5,83 \times 10^{-5}$
activation	relu	tanh

Estas alterações foram baseadas em um processo iterativo de avaliação de desempenho e visam melhorar a precisão e eficiência do modelo. Vale salientar que a função de ativação na camada de saída de um MLPClassifier no scikit-learn, por padrão, é a função de ativação logística (também conhecida como função sigmoid) para problemas binários ou a função softmax para problemas multiclasse.

4.4.3 Matriz de Confusão e Análise de Resultados

A performance do modelo foi também analisada por meio de uma matriz de confusão, que oferece uma visão detalhada do desempenho em classificar corretamente as classes positivas e negativas. A Tabela 17 apresenta a matriz de confusão com totalizadores.

		Classe Predita		Total
		Não-Cancelado	Cancelado	
Classe Real	Não-Cancelado	21.347	1.265	22.612
	Cancelado	1.682	20.162	21.844
Total Predito		23.029	21.427	44.456

Tabela 14 – Matriz de confusão com totalizadores

As métricas recalculadas baseiam-se nos seguintes valores atualizados da matriz de confusão:

- Verdadeiros Positivos (VP): 21.347
- Falsos Negativos (FN): 1.265

- Falsos Positivos (FP): 1.682
- Verdadeiros Negativos (VN): 20.162
- Total Real de Positivos (VP + FN): 22.612
- Total Real de Negativos (FP + VN): 21.844
- Total Predito de Positivos (VP + FP): 23.029
- Total Predito de Negativos (FN + VN): 21.427
- Total Geral (VP + VN + FP + FN): 44.456

Métrica	Fórmula	Resultado
Sensibilidade (Taxa de verdadeiro Cancelado)	$\frac{VC}{VC+FNC}$	$\approx 0,9230$
Especificidade (Taxa de verdadeiro Não-Cancelado)	$\frac{VNC}{VNC+FC}$	$\approx 0,9441$
Acurácia	$\frac{VC+VNC}{VC+VNC+FC+FNC}$	$\approx 0,9337$
Valor Preditivo Cancelado (VPC)	$\frac{VC}{VC+FC}$	$\approx 0,9410$
Valor Preditivo Não-Cancelado (VPNC)	$\frac{VNC}{VNC+FNC}$	$\approx 0,9270$

Tabela 15 – Cálculo das Métricas de Avaliação do Modelo.

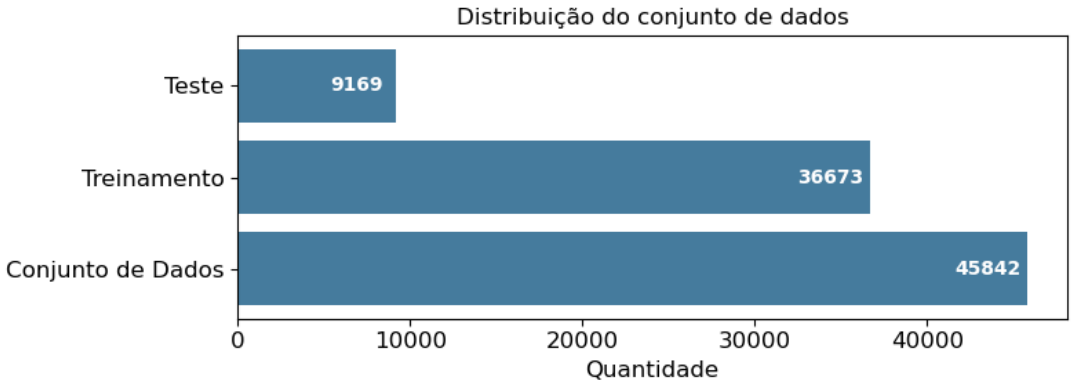
O modelo de classificação apresenta excelentes resultados, com uma sensibilidade de 94,41%, indicando alta eficiência na identificação de casos verdadeiramente positivos. Com uma especificidade de 92,30%, ele também prova ser confiável na exclusão de falsos positivos. A acurácia geral atinge 93,37%, refletindo um desempenho robusto em todas as previsões.

Os valores preditivos positivo e negativo, de 92,70% e 94,10% respectivamente, reforçam a confiança nas classificações feitas pelo modelo. As razões de verossimilhança, tanto positiva (12,26) quanto negativa (0,0606), destacam a capacidade do modelo em fornecer previsões significativas. Esses índices apontam para um modelo que não só é preciso, mas também oferece previsões com alta veracidade, tornando-o uma ferramenta valiosa em contextos onde decisões precisas são críticas.

4.5 Simulações da Análise de Sentimentos

4.5.1 Distribuição dos Dados

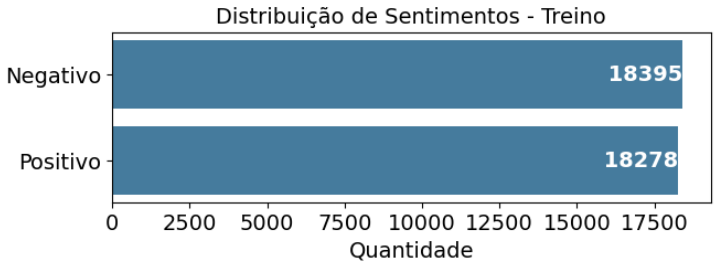
Figura 28 – Distribuição dos dados para análise de sentimentos.



Fonte: O próprio autor

Como podemos verificar na Figura 29, os dados de teste estão devidamente balanceados, isso é importante, principalmente, para que o nosso modelo não tenha uma tedência de aprendizado e, consequentemente, na sua predição.

Figura 29 – Distribuição dos dados para análise de sentimentos.



Fonte: O próprio autor

4.5.2 Hiperparâmetros do Modelo

O modelo de análise de sentimentos foi construído utilizando uma arquitetura de rede neural com camadas de *embedding*, LSTM e densa. A Tabela 16 descreve os hiperparâmetros utilizados para cada camada do modelo.

Tabela 16 – Descrição dos Hiperparâmetros do Modelo

Camada/Hiperparâmetro	**Descrição**	**Valor**
Camada de Embedding	Tamanho do Vocabulário (input_dim)	23305
	Dimensões do Embedding (output_dim)	50
	Comprimento de Entrada (input_length)	104
	Inicializador (embeddings_initializer)	(-0,05, 0,05)
Camada LSTM	Número de Unidades (units)	64
	Dropout (dropout)	0,2
	Dropout Recorrente (recurrent_dropout)	0,2
	Ativação (activation)	tanh
	Ativação Recorrente (recurrent_activation)	sigmoid
Camada Densa	Número de Unidades (units)	2
	Função de Ativação (activation)	softmax

Fonte: Produção do próprio autor

4.5.3 Matriz de Confusão e Análise de Resultados

A performance do modelo foi também analisada por meio de uma matriz de confusão, que oferece uma visão detalhada do desempenho em classificar corretamente as classes positivas e negativas. A Tabela 17 apresenta a matriz de confusão com totalizadores.

		Classe Predita		Total
		Positiva	Negativa	
Classe Real	Positiva	4.511	111	4.622
	Negativa	46	4.501	4.547
Total Predito		4.557	4.612	9.169

Tabela 17 – Matriz de confusão com totalizadores

Estes valores resumem a matriz de confusão e os totalizadores correspondentes:

- Verdadeiros Positivos (VP): 4.511
- Falsos Negativos (FN): 111
- Falsos Positivos (FP): 46
- Verdadeiros Negativos (VN): 4.501
- Total Real de Positivos (VP + FN): 4.622
- Total Real de Negativos (FP + VN): 4.547
- Total Predito de Positivos (VP + FP): 4.557

- Total Predito de Negativos (FN + VN): 4.612
- Total Geral (VP + VN + FP + FN): 9.169

A Tabela 18 ilustra as métricas derivadas da matriz de confusão, incluindo sensibilidade, especificidade, acurácia, valor preditivo positivo e negativo, e as razões de verossimilhança positiva e negativa.

Métrica	Fórmula	Resultado
Sensibilidade (Taxa de verdadeiro positivo)	$\frac{VP}{VP+FN}$	$\approx 0,976$
Especificidade (Taxa de verdadeiro negativo)	$\frac{VN}{VN+FP}$	$\approx 0,990$
Acurácia	$\frac{VP+VN}{VP+VN+FP+FN}$	$\approx 0,983$
Valor Preditivo Positivo (VPP)	$\frac{VP}{VP+FP}$	$\approx 0,990$
Valor Preditivo Negativo (VPN)	$\frac{VN}{VN+FN}$	$\approx 0,976$

Tabela 18 – Cálculo das Métricas de avaliação do modelo.

O modelo de classificação associado à matriz de confusão apresentada demonstra um desempenho altamente eficiente em termos de diferenciar entre as classes positivas e negativas. A sensibilidade de 97,6% indica que o modelo é excepcionalmente bom em detectar casos positivos, enquanto a especificidade de 99,0% mostra que ele também é muito eficaz em identificar os casos negativos corretamente. A acurácia geral de 98,3% reflete uma taxa de sucesso muito alta em todas as previsões feitas pelo modelo.

Adicionalmente, o Valor Preditivo Positivo (VPP) de 99,0% significa que as previsões positivas do modelo quase sempre correspondem à realidade, o que é corroborado por uma Razão de Verossimilhança Positiva (RV+) extremamente alta de 96,5%, sugerindo que um resultado positivo aumenta substancialmente a probabilidade da presença da condição. Por outro lado, o Valor Preditivo Negativo (VPN) de 97,6% e uma Razão de Verossimilhança Negativa (RV-) de 0,0243 confirmam que o modelo é igualmente confiável em prever a ausência da condição.

5 CONCLUSÃO E TRABALHOS FUTUROS

5.1 Conclusão

O objetivo principal deste projeto foi explorar o uso de inteligência artificial para resolver problemas chave em um e-commerce, incluindo recomendação de produtos, otimização de lucro, previsão de cancelamento de pedidos e análise de sentimentos. A metodologia sugerida fundamenta-se principalmente na aplicação de técnicas de aprendizado de máquina, especialmente redes neurais, para lidar com esses complexos desafios. Esta abordagem é motivada pela necessidade de soluções mais eficientes e precisas no crescente campo do comércio eletrônico.

Como resultado, ao utilizar uma metodologia focada em redes neurais e ferramentas como a filtragem colaborativa e o Optuna para otimização de parâmetros, conseguimos desenvolver modelos com alta precisão e eficácia. Em particular, a técnica de filtragem colaborativa aplicada à recomendação de produtos possui uma margem de otimização a ser explorada, como mencionado no capítulo anterior. A otimização de parâmetros de entrada para maximização do lucro demonstrou ser uma estratégia eficaz e inovadora.

A partir dos valores das métricas obtidas, como a `f1_score` de 0.66 no sistema de recomendação, um R^2 de 0.98 na otimização de lucro, uma acurácia de 0.93 na previsão de cancelamentos e 0.98 em análise de sentimentos, fica evidente que a aplicação de redes neurais pode significativamente melhorar diversas operações em e-commerce. Estes resultados reforçam a eficácia das redes neurais na solução de problemas complexos e na geração de insights valiosos para empresas de e-commerce.

Este trabalho contribui significativamente para o campo do aprendizado de máquina aplicada ao e-commerce, demonstrando como técnicas de redes neurais podem ser empregadas para melhorar a eficiência operacional e a satisfação do cliente. Além disso, fornece um caminho para futuras pesquisas na otimização de sistemas de aprendizado de máquina em contextos comerciais, abrindo portas para inovações contínuas no setor.

5.2 Temas a serem pesquisados

- Realizar uma análise comparativa técnicas para avaliar similaridades em sistemas de recomendação de filtragem colaborativa.
- Utilizar redução de dimensionalidade como SVD (Decomposição em valores singulares, do inglês Singular Value Decomposition) para reduzir a dimensionalidade dos dados, potencialmente melhorando o desempenho em conjuntos de dados grandes e esparsos.
- Desenvolver e testar sistemas híbridos para resolver o problema de 'cold start' em sistemas de Recomendação.
- Aprimorar a precisão na previsão de cancelamentos de pedidos investigado o uso da análise de sentimentos como um parâmetro de entrada do modelo.
- Realizar análise de Sentimentos com três saídas incluindo o sentimento "neutro".

REFERÊNCIAS

AGGARWAL, C. C. Neural Networks and Deep Learning: A Textbook. [S.l.]: Springer Cham, 2018. Citado 2 vezes nas páginas 21 e 22.

ALAMDARI, P. M.; NAVIMIPOUR, N. J.; HOSSEINZADEH, M.; SAFAEI, A. A.; DARWESH, A. A systematic study on the recommender systems in the e-commerce. IEEE Access, v. 8, p. 115694–115716, 2020. Citado na página 27.

ALMONTE, L.; GUERRA, E.; CANTADOR, I. et al. Recommender systems in model-driven engineering. Software and Systems Modeling, Springer, v. 21, p. 249–280, 2022. Disponível em: <https://doi.org/10.1007/s10270-021-00905-x>. Citado na página 29.

BHASKAR, J.; SRUTHI, K.; NEDUNGADI, P. Hybrid approach for emotion classification of audio conversation based on text and speech mining. Procedia Computer Science, v. 46, p. 635–643, 2015. ISSN 1877-0509. Proceedings of the International Conference on Information and Communication Technologies, ICICT 2014, 3-5 December 2014 at Bolgatty Palace Island Resort, Kochi, India. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1877050915001763>. Citado na página 30.

BISHOP, C. M. Pattern Recognition and Machine Learning. New York: Springer, 2006. Citado 3 vezes nas páginas 14, 19 e 20.

BLOOMBERG. Alibaba beats estimates as personalised recommendations boost sales. 2019. [cited 2021 23 January 2021]. Disponível em: <https://www.businessoffashion.com/articles/china/alibaba-beats-estimates-as-recommendations-boost-sales>. Citado na página 26.

FORTMANN-ROE, S. Understanding the Bias-Variance Tradeoff. 2012. Acesso em: 27 jul. 2023. Disponível em: <https://scott.fortmann-roe.com/docs/BiasVariance.html>. Citado na página 21.

GROSSE, R. CS321 Lecture Notes: Unsupervised Learning. 2023. Acesso em: 27 jul. 2023. Disponível em: <https://www.cs.toronto.edu/>. Citado na página 16.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Second. [S.l.]: Springer, 2009. (Springer Series in Statistics). Citado 2 vezes nas páginas 18 e 19.

HAYKIN, S. S. Neural Networks and Learning Machines. 3. ed. [S.l.]: Prentice Hall, 2009. Citado 3 vezes nas páginas 21, 22 e 23.

KEIKHOSROKIANI, P.; FYE, G. A hybrid recommender system for health supplement e-commerce based on customer data implicit ratings. Multimed Tools Appl, 2023. Disponível em: <https://doi.org/10.1007/s11042-023-17321-6>. Citado 2 vezes nas páginas 27 e 29.

KRATZWALD, B.; ILIć, S.; KRAUS, M.; FEUERRIEGEL, S.; PRENDINGER, H. Deep learning for affective computing: Text-based emotion recognition in decision support. Decision Support Systems, v. 115, p. 24–35, 2018. ISSN 0167-9236. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167923618301519>. Citado na página 30.

LAKSHMI, T. J.; BHAVANI, S. D. Link prediction in temporal heterogeneous networks. In: WANG, G.; CHAU, M.; CHEN, H. (Ed.). Intelligence and Security Informatics. Springer, Cham, 2017. (Lecture Notes in Computer Science, v. 10241). ISBN 978-3-319-57462-2. Disponível em: <https://doi.org/10.1007/978-3-319-57463-9_6>. Citado na página 28.

LIMA, B. D. C. Previsão de Cancelamento de Assinaturas de Vinho com Técnicas de Aprendizado de Máquina. 2021. [Trabalho de Conclusão de Curso, Universidade Federal do Espírito Santo]. Disponível em: <https://ele.ufes.br/pt-br/projetos-de-graduacao-202101-e-202102>. Citado na página 32.

LIU, F.; ZHENG, J.; ZHENG, L.; CHEN, C. Combining attention-based bidirectional gated recurrent neural network and two-dimensional convolutional neural network for document-level sentiment classification. Neurocomputing, v. 371, p. 39–50, 2020. ISSN 0925-2312. Disponível em: <https://www.sciencedirect.com/science/article/pii/S092523121931272X>. Citado na página 30.

MANGALINDAN, J. Amazon's recommendation secret. 2012. [cited 2021 23 January 2021]. Disponível em: <https://fortune.com/2012/07/30/amazons-recommendation-secret/>. Citado na página 26.

MITCHELL, T. M. Machine Learning. [S.l.: s.n.], 1997. Citado 4 vezes nas páginas 17, 20, 25 e 33.

MURPHY, K. P. Machine Learning: A Probabilistic Perspective. 1. ed. [S.l.]: MIT Press, 2012. Citado na página 17.

NANDWANI, P.; VERMA, R. A review on sentiment analysis and emotion detection from text. Social Network Analysis and Mining, Springer, v. 11, p. 81, 2021. Disponível em: <https://doi.org/10.1007/s13278-021-00776-6>. Citado 2 vezes nas páginas 30 e 31.

ROLIM, V.; FERREIRA, R.; COSTA, E.; PINHEIRO, A.; FERREIRA, M. A. Um estudo sobre sistemas de recomendação de recursos educacionais. p. 724, 10 2017. Citado 2 vezes nas páginas 28 e 29.

RUSSEL, S.; NORVIG, P. Artificial Intelligence: A Modern Approach. Second. [S.l.]: Prentice-Hall, 2009. Citado 3 vezes nas páginas 15, 16 e 18.

SHARMA, L.; GERA, A. A survey of recommendation system: Research challenges. International Journal of Engineering Trends and Technology (IJETT), v. 4, n. May, 2013. Faridabad. Citado 2 vezes nas páginas 28 e 29.

SU, X.; KHOSHGOFTAAR, T. M. A survey of collaborative filtering techniques. Nome da Revista, Florida Atlantic University, Boca Raton, FL, 2009. Available from: <https://downloads.hindawi.com/archive/2009/421425.pdf>. Citado na página 29.

TAULLI, T. Artificial Intelligence Basics: A Non-Technical Introduction. [S.l.]: Apress, 2019. Citado 4 vezes nas páginas 14, 15, 26 e 33.

WU, L.; HOI, S. C. H.; YU, N. Semantics-preserving bag-of-words models and applications. IEEE Transactions on Image Processing, v. 19, n. 7, p. 1908–1920, 2010. Citado na página 30.