UNIVERSIDADE FEDERAL DE SANTA CATARINA
CAMPUS ARARANGUÁ
CENTRO DE CIÊNCIAS, TECNOLOGIAS E SAÚDE (CTS)
TECNOLOGIAS DA INFORMAÇÃO E COMUNICAÇÃO (TIC)

IMPLEMENTAÇÃO DE UMA INTERFACE GRÁFICA PARA USO DE ALGORITMOS DE APRENDIZADO DE MÁQUINA

Vinicius Rover

Araranguá
2024

VINÍCIUS ROVER

IMPLEMENTAÇÃO DE UMA INTERFACE GRÁFICA PARA USO DE ALGORITMOS DE APRENDIZADO DE MÁQUINA

Trabalho de Conclusão de Curso submetido ao curso em Tecnologias da Informação e Comunicação do Campus Araranguá da Universidade Federal de Santa Catarina como requisito parcial para a obtenção do título de Bacharel em Tecnologias de Informação e Comunicação.

Orientador(a): Prof. Anderson Luiz Fernandes Perez, Dr.

Araranguá

2024

Rover, Vinicius
Implementação De Uma Interface Gráfica para uso de Algoritmos de aprendizado de máquina / Vinicius Rover ; orientador, Anderson Luiz Fernandes Perez, 2024.
46 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Campus Araranguá, Graduação em Tecnologias da Informação e Comunicação, Araranguá, 2024.

Inclui referências.

1. Tecnologias da Informação e Comunicação. 2. Desenvolvimento Web. 3. Aprendizado de Máquina. 4. Interface Gráfica. I. Perez, Anderson Luiz Fernandes. II. Universidade Federal de Santa Catarina. Graduação em Tecnologias da Informação e Comunicação. III. Título.

Vinicius Rover

IMPLEMENTAÇÃO DE UMA INTERFACE GRÁFICA PARA USO DE ALGORITMOS DE APRENDIZADO DE MÁQUINA

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do título de Bacharel em Tecnologias de Informação e Comunicação e aprovado em sua forma final pelo Curso Tecnologias de Informação e Comunicação.

Araranguá, 27 de Junho de 2024.

Documento assinado digitalmente
Fernando Jose Spanhol
Data: 10/07/2024 20:05:21-0300
CPF: ***.656.419-**
Verifique as assinaturas em https://v.ufsc.br

Coordenação do Curso

Banca examinadora

Documento assinado digitalmente
Anderson Luiz Fernandes Perez
Data: 10/07/2024 18:15:28-0300
CPF: ***.803.459-**
Verifique as assinaturas em https://v.ufsc.br

Prof.(a) Anderson Luiz Fernandes Perez, Dr.(a)
Orientador(a)

Documento assinado digitalmente
Marina Carradore Sergio
Data: 10/07/2024 21:03:34-0300
CPF: ***.746.359-**
Verifique as assinaturas em https://v.ufsc.br

Prof.(a) Marina Carradore Sérgio, Dr.(a)
Universidade Federal de Santa Catarina

Documento assinado digitalmente
Fabricio Herpich
Data: 10/07/2024 22:58:48-0300
CPF: ***.335.260-**
Verifique as assinaturas em https://v.ufsc.br

Prof.(a) Fabrício Herpich, Dr.(a)
Universidade Federal de Santa Catarina

Araranguá, 2024

AGRADECIMENTOS

Aos meus pais, por acreditarem em mim e me apoiarem incondicionalmente em todos os momentos da minha vida. Seu amor, paciência e incentivo foram fundamentais para que eu chegasse até aqui.

Ao meu orientador, Dr. Anderson Luiz Fernandes Perez, por sua orientação, paciência e por compartilhar seu conhecimento e experiência. Suas valiosas contribuições foram essenciais para a realização deste trabalho. Agradeço também pela confiança e pelo suporte em todas as etapas do projeto.

Aos professores do campus Araranguá, por todo o conhecimento transmitido ao longo dos anos e pela dedicação à formação dos alunos. Suas aulas e orientações foram fundamentais para o meu desenvolvimento acadêmico.

Aos meus colegas e amigos, que compartilharam comigo os momentos de alegria e dificuldades durante o curso.

Aos profissionais e colaboradores da UFSC, pela disponibilidade em compartilhar informações e por contribuírem diretamente com a realização deste estudo.

A todos que, de forma direta ou indireta, contribuíram para a minha formação, minha sincera gratidão.

RESUMO

O presente trabalho tem como objetivo o desenvolvimento de uma aplicação web destinada a facilitar o uso de algoritmos de aprendizado de máquina. A interface do usuário foi construída utilizando React, uma biblioteca JavaScript popular para a criação de interfaces dinâmicas e responsivas. Para o treinamento e a implementação dos modelos de aprendizado de máquina, foram utilizados Python, a biblioteca scikit-learn e o framework FastAPI, que permite a criação de uma API eficiente e de alta performance. A aplicação web proporciona aos usuários uma experiência interativa e educativa, permitindo a visualização e a experimentação com diferentes algoritmos de aprendizado de máquina. Os principais algoritmos abordados incluem regressão linear, árvores de decisão, máquinas de vetores de suporte (SVM), redes neurais artificiais, entre outros. A relevância deste projeto reside na crescente demanda por profissionais capacitados em aprendizado de máquina e na necessidade de ferramentas que tornem esse conhecimento acessível e compreensível. A aplicação proposta visa suprir essa lacuna, oferecendo um recurso valioso tanto para estudantes quanto para profissionais que desejam aprofundar seus conhecimentos na área. Este trabalho abre caminho para futuras melhorias e expansões, como a inclusão de mais algoritmos, a adição de funcionalidades avançadas de visualização de dados, e a implementação de técnicas mais complexas de aprendizado de máquina. Além disso, a aplicação pode evoluir para suportar diferentes formatos de dados e novos paradigmas de aprendizado, como aprendizado profundo e aprendizado por reforço.

Palavras-chave: Aprendizado de máquina, Desenvolvimento Web, interface gráfica

ABSTRACT

This project aims to develop a web application designed to facilitate the teaching of machine learning through the use of various algorithms. The user interface was built using React, a popular JavaScript library for creating dynamic and responsive interfaces. For training and implementing the machine learning models, Python, the scikit-learn library, and the FastAPI framework will be used, enabling the creation of an efficient and high-performance API. The web application provides users with an interactive and educational experience, allowing them to visualize and experiment with different machine learning algorithms. The main algorithms covered include linear regression, decision trees, support vector machines (SVM), neural networks, among others. The relevance of this project lies in the growing demand for professionals skilled in machine learning and the need for educational tools that make this knowledge accessible and comprehensible. The proposed application aims to fill this gap, offering a valuable resource for both students and professionals seeking to deepen their understanding in the field. This project paves the way for future improvements and expansions, such as the inclusion of more algorithms, the addition of advanced data visualization features, and the implementation of more complex machine learning techniques. Additionally, the application can evolve to support different data formats and accommodate new learning paradigms, such as deep learning and reinforcement learning.

Keywords: Machine Learning, Web Development, graphic interface

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

A implementação de uma Interface Gráfica (GUI) para o uso de algoritmos de Aprendizado de Máquina é uma abordagem essencial para tornar essa tecnologia mais acessível e prática para um público mais amplo. Enquanto o Aprendizado de Máquina desempenha um papel cada vez mais significativo em diversas aplicações, desde análise de dados até automação de tarefas, muitos usuários enfrentam barreiras técnicas para explorar seu potencial.

Uma das grandes vantagens das GUIs é a redução da curva de aprendizado. Novos usuários podem rapidamente começar a experimentar e implementar algoritmos sem passar meses ou anos aprendendo linguagens de programação complexas e técnicas matemáticas avançadas. Isso é especialmente benéfico em ambientes educacionais, onde os estudantes podem visualizar e manipular dados de maneira interativa, aumentando seu entendimento e interesse na área.

Além disso, as GUIs facilitam a colaboração interdisciplinar. Em muitas organizações, especialistas de diferentes áreas precisam trabalhar juntos para resolver problemas complexos. Interfaces gráficas permitem que profissionais de marketing, finanças, medicina e outros campos colaborem com cientistas de dados de maneira mais eficiente, pois não é necessário que todos tenham habilidades avançadas em programação para contribuir com o processo de análise de dados.

Outra vantagem significativa é a aceleração do processo de prototipagem e desenvolvimento. Em um ambiente de negócios competitivo, a capacidade de desenvolver e testar rapidamente novos modelos pode ser um diferencial importante. GUIs permitem que as equipes de desenvolvimento ajustem parâmetros e testem diferentes algoritmos de maneira mais ágil, respondendo rapidamente às mudanças nas necessidades do mercado.

De acordo com Silva et al. (2019), “a democratização do aprendizado de máquina através de interfaces gráficas pode impulsionar inovações em diversas áreas, tornando essa tecnologia mais acessível e amigável para todos”. Isso também inclui a possibilidade de pequenas empresas e startups se beneficiarem de

tecnologias avançadas sem a necessidade de contratar especialistas caros, nivelando o campo de competição com empresas maiores e mais estabelecidas.

A combinação de Aprendizado de Máquina e Interfaces Gráficas oferece a oportunidade de democratizar essa tecnologia e impulsionar inovações em diversas áreas. Como aponta Santos (2018), “essa abordagem não só amplia o acesso, mas também promove uma maior inclusão digital, permitindo que mais pessoas possam utilizar e beneficiar-se das ferramentas avançadas de análise de dados”. Portanto, a implementação de GUIs é uma estratégia crucial para ampliar o alcance e o impacto do aprendizado de máquina na sociedade contemporânea.

Finalmente, a personalização é outra área onde as GUIs podem agregar valor significativo. Usuários podem customizar suas interações com os algoritmos de aprendizado de máquina, ajustando as interfaces para melhor atender às suas necessidades específicas e preferências. Isso pode aumentar a eficiência e a eficácia das soluções de aprendizado de máquina, tornando-as mais adaptáveis e responsivas às necessidades individuais dos usuários.

Em resumo, a implementação de GUIs para algoritmos de Aprendizado de Máquina não só torna essa tecnologia mais acessível e prática, mas também promove a colaboração interdisciplinar, acelera o desenvolvimento de protótipos, democratiza o acesso a tecnologias avançadas e permite a personalização. Esses benefícios juntos tornam as GUIs uma ferramenta poderosa para a disseminação e aplicação do aprendizado de máquina em uma ampla gama de contextos.

1. 1 OBJETIVOS

Esta seção apresenta o objetivo geral e os objetivos específicos deste trabalho de conclusão de curso.

1.1.1 Objetivo geral

O objetivo geral do presente trabalho foi criar uma interface gráfica que propicie o uso de algumas técnicas de aprendizado de máquina de uma forma mais simplificada.

1.1.2 Objetivos específicos

Com base no objetivo geral proposto são especificados os seguintes objetivos específicos:

- Criar um design intuitivo e esteticamente agradável para a interface do usuário;

- Implementar a parte visual da interface gráfica conforme projetado;

- Desenvolver uma API funcional capaz de receber e processar arquivos de entrada e executar algoritmos de aprendizado de máquina.

1.2 JUSTIFICATIVA

A área de aprendizado de máquina enfrenta uma significativa falta de profissionais qualificados, um fenômeno que se tornou um dos maiores desafios para a expansão e inovação tecnológica em diversas indústrias. A *World Economic Forum* estima um crescimento de 40% nas vagas relacionadas à IA até 2025, destacando a necessidade de especialistas em machine learning, analistas de dados e cientistas de dados

Primeiramente, a velocidade com que a tecnologia avança supera a capacidade das instituições educacionais de formar profissionais preparados. Segundo um relatório da McKinsey & Company (2021), "a demanda por habilidades

de inteligência artificial (IA) e aprendizado de máquina (AM) está crescendo tão rapidamente que os sistemas educacionais estão lutando para acompanhar".

Além disso, a complexidade inerente ao aprendizado de máquina exige uma combinação de habilidades em matemática, estatística e ciência da computação, áreas onde tradicionalmente há um déficit de interesse e, consequentemente, de profissionais. De acordo com um estudo realizado pela IBM (2019), "a maioria dos cargos de IA e AM exige um conjunto especializado de habilidades que são desenvolvidas apenas através de programas educacionais avançados e experiência prática, o que restringe o pool de candidatos qualificados".

Outro fator é a alta competitividade no mercado de trabalho. Profissionais capacitados em aprendizado de máquina são altamente cobiçados não apenas por empresas de tecnologia, mas também por setores como finanças, saúde e manufatura, aumentando a disputa por talentos e dificultando a retenção de profissionais. Conforme observado pela Gartner (2022), "as empresas estão investindo pesadamente para atrair e reter talentos em aprendizado de máquina, oferecendo pacotes de compensação atraentes e oportunidades de desenvolvimento de carreira, o que eleva o nível de competitividade no setor".

Por fim, a falta de programas de treinamento e desenvolvimento contínuo nas empresas também contribui para essa escassez. Muitas organizações ainda não implementaram iniciativas eficazes para formar internamente seus próprios especialistas em aprendizado de máquina. Um relatório da Deloitte (2020) destaca que "a capacitação contínua e a criação de pipelines internos de talentos são cruciais para atender à demanda crescente, mas muitas empresas ainda estão nos estágios iniciais de desenvolvimento dessas estratégias".

Portanto, é evidente que a falta de profissionais na área de aprendizado de máquina é um problema multifacetado que requer abordagens integradas envolvendo educação. Por isso, a busca por soluções que facilitem o acesso ao aprendizado de máquina tem se tornado uma prioridade no campo da inteligência artificial. Nesse contexto, a criação de uma Interface Gráfica (GUI) ganha destaque, pois seu objetivo é simplificar significativamente o acesso a técnicas de aprendizado de máquina, tornando-as mais acessíveis a um público amplo.

O diferencial dessa GUI reside na sua capacidade de permitir que pessoas com pouco conhecimento em Inteligência Artificial possam utilizar algoritmos de aprendizado de máquina com facilidade e eficácia, sem a necessidade de uma compreensão profunda das complexidades matemáticas subjacentes.

Isso significa que qualquer pessoa, independentemente de sua formação acadêmica ou experiência anterior na área, poderá explorar e experimentar com diferentes modelos de aprendizado de máquina de forma descomplicada e de qualquer lugar.

A GUI desenvolvida visa a minimização das barreiras técnicas que tradicionalmente afastam potenciais interessados no aprendizado de máquina. Essa abordagem permitirá aos usuários aplicar técnicas de aprendizado de máquina em diversas áreas, como análise de dados, reconhecimento de padrões e tomada de decisões automatizadas, contribuindo assim para a resolução de problemas do mundo real.

Portanto, GUI proposta neste TCC se destina a fornecer uma experiência amigável e intuitiva aos usuários, permitindo que eles treinem, avaliem e apliquem modelos de aprendizado de máquina com facilidade.

1.3 METODOLOGIA

A metodologia adotada para o presente trabalho envolve uma abordagem integrada que combina pesquisa bibliográfica e desenvolvimento de software. O escopo do projeto é a "Implementação de uma Interface Gráfica para Uso de Algoritmos de Aprendizado de Máquina", o que requer uma estratégia que permita tanto a compreensão aprofundada do tema quanto a aplicação prática dos conhecimentos adquiridos.

1.3.1 Pesquisa Bibliográfica

A pesquisa bibliográfica é alicerçada na revisão exaustiva da literatura disponível relacionada ao uso de algoritmos de aprendizado de máquina e à criação de interfaces gráficas amigáveis para usuários leigos na área. Este método proporcionará uma compreensão sólida dos conceitos teóricos, frameworks existentes e melhores práticas relacionadas ao tema. A metodologia bibliográfica abrange:

- Identificação de Fontes Relevantes: localização e seleção de livros, artigos científicos, documentos técnicos e recursos online pertinentes à implementação de interfaces gráficas para algoritmos de aprendizado de máquina.

- Análise Crítica: avaliação crítica das fontes selecionadas para extrair conceitos-chave, tendências, desafios e abordagens inovadoras na integração de algoritmos de aprendizado de máquina em interfaces gráficas.

- Sistematização do Conhecimento: organização e síntese dos conhecimentos adquiridos para estabelecer uma base teórica sólida que orientará o desenvolvimento prático.

1.3.2 Desenvolvimento de Software

A etapa de desenvolvimento se concentra na materialização dos conceitos teóricos em uma aplicação prática. A metodologia adotada para o desenvolvimento de software envolve:

- Design da Interface Gráfica: utilização de ferramentas de prototipagem para criar modelos iniciais da interface, considerando princípios de usabilidade e acessibilidade para usuários leigos em aprendizado de máquina.

- Escolha de Tecnologias: seleção criteriosa de tecnologias de desenvolvimento de software adequadas ao escopo do projeto, justificando as escolhas com base em critérios de eficiência, escalabilidade e compatibilidade.

- Integração de Algoritmos: desenvolvimento de uma API funcional capaz de interagir com algoritmos de aprendizado de máquina, permitindo sua execução a partir da interface gráfica.

- Testes e Avaliação: implementação de testes de usabilidade e funcionalidade para garantir a eficácia da interface gráfica e a correta integração com os algoritmos.

- Iteração e Ajustes: ciclos iterativos de feedback, promovendo ajustes na interface com base nas avaliações dos usuários, garantindo a melhoria contínua do software.

Essa abordagem metodológica visa combinar a teoria sólida adquirida por meio da pesquisa bibliográfica com a aplicação prática na forma de uma interface gráfica funcional para algoritmos de aprendizado de máquina, contribuindo para a construção de um trabalho abrangente e significativo no contexto proposto.

1.4 Organização do Trabalho

Este trabalho de conclusão de curso, além desta introdução, está organizado em mais 4 capítulos. O segundo capítulo apresenta os conceitos fundamentais do

aprendizado de máquina, destacando sua importância e aplicações em diversas áreas. Este capítulo também descreve os diferentes tipos de algoritmos de aprendizado de máquina e suas diferenças, como algoritmos supervisionados, não supervisionados, semi-supervisionados e de reforço. No terceiro capítulo são descritos os algoritmos de aprendizado de máquina implementados neste trabalho. São abordados os seguintes algoritmos: Árvore de Decisão, Support Vector Machines (SVM), Redes Neurais Artificiais (ANN), Árvore de Regressão, Redes Neurais Artificiais em problemas de regressão, Stochastic Gradient Descent (SGD), K-means e DBSCAN. O quarto capítulo apresenta o projeto do sistema desenvolvido neste trabalho. Inclui uma descrição detalhada da entrada de dados, a estrutura e os componentes da tela desenvolvida em React, e o desenvolvimento da API que suporta o sistema, detalhando a estrutura da API e as tecnologias utilizadas. O quinto capítulo apresenta as considerações finais e propostas para melhorias futuras.

2 Aprendizado de Máquina

O aprendizado de máquina refere-se ao estudo de algoritmos que se adaptam a problemas de forma automatizada. Este tipo de aprendizagem tem sido utilizado em aplicações que vão desde a mineração de dados até sistemas de filtragem de informações (LUDERMIR, 2021).

Algoritmos de ML também são usados para identificar páginas web maliciosas. Siqueira, et al, (2021) relacionaram o aprendizado de máquina como um conjunto de métodos que podem ser executados automaticamente, detectar padrões em dados e usar os padrões descobertos para prever dados futuros ou para realizar outros tipos de tomada de decisão sob incerteza, como, por exemplo, planejar como coletar mais dados.

Para Ludermir (2021) as técnicas de mineração de dados e aprendizado de máquina utilizam modelos probabilísticos eficientes como: modelos de regressão generalizada, redes neurais artificiais, árvores de decisão e redes de crenças Bayesianas para determinar e prever ou pontuar uma fraude, por exemplo. Dados das transações realizadas pelos clientes são utilizados para determinar os padrões, e estes, permitem identificar rapidamente circunstâncias fora do comportamento diário de um cliente que podem ser indícios de fraude.

A mineração de dados é o processo de descoberta de padrões interessantes a partir de grandes quantidades de dados, como um processo de descoberta de informações potencialmente úteis e novas. Padrões interessantes representam conhecimento e medidas de interesse no padrão, sejam objetivos ou subjetivos, e podem ser usados para orientar o processo de descoberta (LUDERMIR, 2021).

Em termos gerais, para Siqueira, et al (2021) ML é uma disciplina híbrida entre Estatística e Ciência da Computação na qual diferentes técnicas e algoritmos são utilizados com o objetivo de gerar sistemas que sejam capazes de realizar previsões e/ou tomar decisões por si próprios com base em dados, diz-se que os algoritmos são treinados nos dados para aprender a tarefa que se deseja.

As técnicas utilizadas podem ser classificadas de várias maneiras. Por exemplo: em problemas de regressão a saída do sistema é um valor previsto para uma variável contínua. Por outro lado, em problemas de classificação, duas ou mais

classes estão disponíveis e a saída é uma classe prevista para cada dado sobre o qual é avaliado.

Em problemas de classificação, é comum chamar os algoritmos treinados de classificadores de dados. Outra distinção baseada nos dados é se alguém tem ou não acesso à saída esperada para as instâncias de treinamento, por exemplo, a classe à qual cada item de dados pertence em problemas de classificação (LUDERMIR, 2021).

Em ML os computadores ou máquinas aprendem sem terem sido explicitamente programados (SANTOS, 2019). De acordo com Santos:

> O aprendizado de máquinas tem como objetivo a execução de uma ou mais tarefas por uma máquina ou programa, em qualquer circunstância que envolva a execução, sem que essa máquina ou computador tenha sido programado para realizar essa tarefa especificamente. Como a máquina e/ou o programa de computador não foi programado para realizar tal tarefa, entende-se que o mesmo aprendeu a realizar a atividade. O uso de aprendizado de máquinas tem crescido a cada dia e cresce o uso de aplicações com base em aprendizado de máquinas. (SANTOS, 2019, p.40)

As aplicações de ML são muito diversas. Pode ocorrer por predição, em que as variáveis de saída são obtidas de acordo com modelos previamente aprendidos, como por exemplo, para prever o clima. Poderá ocorrer também por mineração de dados, que consiste em uma das mais importantes, onde o objetivo é extrair padrões de informações relevantes de grandes conjuntos de dados (SIQUEIRA; et al, 2021).

Ainda de acordo com Santos (2019) também se dá por reconhecimento de imagem, como por exemplo, rosto e imagens; por análise do comportamento do consumo e da produtividade, onde analisa os clientes e suas dúvidas para oferecer o produto que possa interessá-los.

Normalmente a construção de modelos é suportada pela disponibilidade de um conjunto de dados de treino, que normalmente consiste num conjunto de padrões que é a unidade para realizar a tarefa de previsão, recomendação, etc. Dependendo se a saída desejada do modelo está disponível ou não para os referidos dados de treinamento, os algoritmos de ML podem ser divididos em: algoritmos com aprendizagem supervisionada e algoritmos com aprendizagem não supervisionada (BISHOP, 2006).

No aprendizado supervisionado, os dados são rotulados no conjunto de dados de treinamento Já no processo de aprendizagem não supervisionado, os exemplos de entrada não são rotulados por classe. Normalmente, é possível usar clustering para descobrir a relação entre os dados de entrada (MAHESH, 2020).

2.1 Abordagens de Treinamento em Algoritmos de Aprendizado de Máquina

a) Aprendizagem Supervisionada

Os algoritmos são treinados com dados rotulados, o que significa que é fornecido o par entrada e saída desejada. Esses algoritmos são usados quando se deseja mapear um conjunto de entradas para um conjunto de saídas conhecidas (HASTIE; TIBSHIRANI; FRIEDMAN, 2009).

> No aprendizado supervisionado, são apresentadas ao sistema de aprendizado várias amostras de características de entrada (variáveis independentes) e suas respectivas saídas (alvo/variável dependente). O objetivo é aprender a regra e/ou relação que melhor mapeia as entradas (características) para as suas respectivas saídas (alvos). Normalmente, esse aprendizado é alcançado através de uma primeira etapa, chamada treinamento, onde uma parte dos dados é apresentada ao sistema de aprendizado para que o mesmo possa, pela modelagem matemática dessas informações, obter a melhor função que relaciona as entradas com as saídas informadas (SANTOS, 2019, p. 26-27).

A aprendizagem supervisionada é caracterizada pelo uso de um conjunto de dados denominado conjunto de treinamento. Assim, é utilizado um conjunto de tuplas $\{(X_1,Y_1),...,(X_n,Y_n)\}$ onde o conjunto $\{X_i\}$, onde $i = 1,...,n$ são os dados de entrada e o conjunto $\{Y_i\}$, onde $i = 1,...,n$ são os dados de saída esperados (LUDERMIR, 2021).

A aprendizagem supervisionada é dividida em dois tipos principais: classificação e regressão. Algoritmos de classificação podem, por exemplo, atribuir *leads* a uma posição em seu funil de vendas (BRAGA; PRATES, 2020).

Algoritmos de regressão podem identificar valores de corte para uma métrica de decisão, como concessão de crédito, com base em uma combinação de fatores (LUDERMIR, 2021).

Algoritmos de aprendizagem supervisionada criam um modelo preditivo que encontra relações entre observações que dizem aos humanos coisas que eles não sabiam sobre os dados que possuem. Mas como a rotulagem de dados é demorada e cara, os algoritmos de aprendizado de máquina supervisionado muitas vezes não possuem todos os dados necessários para atingir seu potencial de previsão. (FONTANA, 2020).

b) Aprendizagem Não-Supervisionada

Algoritmos não supervisionados são usados para explorar a estrutura oculta nos dados quando não há rótulos ou saídas desejadas associadas a eles. Esses algoritmos são frequentemente usados para tarefas de agrupamento, redução de dimensionalidade e extração de padrões dos dados (BISHOP, 2006).

Os algoritmos supervisionados são usados quando se tem dados rotulados e deseja fazer previsões ou classificações com base nesses rótulos, enquanto algoritmos não supervisionados são aplicados quando se deseja explorar a estrutura e padrões em seus dados sem orientação de rótulos.

> Ao contrário do aprendizado supervisionado, no aprendizado não supervisionado, nenhuma saída é dada para o sistema de aprendizado. Nesse caso, o sistema de aprendizado tenta encontrar uma estrutura, similaridade ou correlação nas entradas fornecidas. Ou seja, no aprendizado não supervisionado, não existem saídas certas ou erradas. Ao invés disso, o objetivo é encontrar um padrão que relaciona as diversas amostras de entrada. Dessa forma, normalmente, o algoritmo de aprendizado de máquina não supervisionado tem como objetivo descobrir novos padrões nos dados ou realizar o agrupamento de entradas, sendo muito utilizados para tarefas de clusterização (SANTOS, 2019, p.28)

Ambos os tipos de algoritmos são fundamentais na aprendizagem de máquina e têm uma ampla gama de aplicações em campos como reconhecimento de padrões, análise de dados, processamento de linguagem natural e muito mais (LUDERMIR, 2021).

A aprendizagem não supervisionada encontra relacionamentos em dados não rotulados. Esse tipo de algoritmo de ML não é capaz de descobrir o que seus dados necessariamente descrevem, mas podem dizer quais observações são semelhantes (SIQUEIRA, et. al. 2021).

No entanto, o enorme volume de dados numa imagem pode tornar o processamento tão lento que a quantidade de informações úteis nos dados é limitada pelo fluxo de dados no sistema. Como resultado, a análise pode não atingir uma pontuação F1 adequada (LUDERMIR, 2021).

A pontuação F1 é um cálculo da proporção de verdadeiros positivos (observações que são classificadas corretamente) em relação ao total de falsos positivos (observações classificadas como algo que não eram) mais falsos negativos (observações não classificadas como algo que eram) (BRAGA; PRATES, 2020).

c) Aprendizagem por Reforço

O aprendizado por reforço permite planejar estratégias eficazes com base na experimentação de dados. Esta é uma forma de otimização baseada em dados. (LUDERMIR, 2021).

A máquina aprende com a própria experiência, interagindo com o ambiente até encontrar o comportamento ideal. Com base na informação disponível, realizará ações que irá repetir e “reforçar” em função da recompensa que obtiver, que podem ser positivas ou negativas (SIQUEIRA, et. al. 2021).

> Neste tipo de sistema, um agente realiza uma ação (dentre uma série de ações possíveis) em um ambiente e recebe uma recompensa de acordo com o resultado dessa ação, sendo o objetivo do algoritmo receber a maior recompensa possível. Estes sistemas possuem três componentes principais: o agente, o ambiente e a forma de interação entre estes dois. O agente é o programa que está sendo treinado. De

> alguma forma, este agente precisa observar, interagir e modificar o ambiente ao longo do tempo. As etapas envolvidas costumam seguir a seguinte sequência:
>
> 1. O agente faz uma observação do ambiente;
>
> 2. O agente escolhe uma ação dentre diversas ações possíveis baseado na observação;
>
> 3. O agente entra em um estado de espera para que o ambiente envie novas observações;
>
> 4. O ambiente executa a ação recebida pelo agente e envia para o agente uma recompensa e a nova configuração do ambiente;
>
> 5. Repete-se as etapas 1-4. (FONTANA, 2020, p.04-05).

Em vez de tomar decisões ou fazer previsões, o aprendizado por reforço gera estratégias automaticamente. Isto permite, entre outras aplicações, a manutenção preditiva ou a personalização das experiências do cliente. Uma das características fundamentais dos sistemas de aprendizagem por reforço é a iniciativa (LUDERMIR, 2021).

É o próprio sistema que explora os dados, analisando as estratégias que funcionaram no passado e repetindo-as quando surgirem situações semelhantes no futuro. Além disso, explora novas ações ou estratégias, avaliando sua eficácia e conseguindo assim melhorar as anteriores e se preparar para novos cenários (BRAGA; PRATES, 2020).

Com essas informações, é possível planejar uma nova estratégia de negócios. O aprendizado por reforço profundo é o campo que unifica o aprendizado profundo e o aprendizado por reforço para criar estratégias complexas que otimizam processos a partir de dados não estruturados, como imagens ou textos (LUDERMIR, 2021).

d) Aprendizagem Semi-Supervisionada

A aprendizagem semi supervisionada aborda problemas onde em um conjunto de dados de treinamento, eventualmente, alguns dados não foram rotulados. Segundo Ludermir (2021) é uma técnica para treinar um computador para resolver um problema, fornecendo exemplos com a resposta correta (dados rotulados) e exemplos sem a resposta correta (dados não rotulados). Esta técnica

trata os pontos de dados de forma diferente dependendo se eles possuem ou não um rótulo.

Se um ponto de dados for rotulado, o algoritmo utiliza o ponto de dados para atualizar os pesos dados aos coeficientes, por exemplo, em uma equação de regressão linear. Se o ponto de dados não estiver rotulado, procura minimizar as diferenças detectadas, por exemplo, na análise k-médias (BRAGA; PRATES, 2020).

A eficácia da aprendizagem semi supervisionada pode ser aumentada por uma coleção de algoritmos de aprendizagem ativa. Algoritmos de aprendizado ativo precisam de menos consultas para obter alta precisão do que a seleção aleatória de consultas (LUDERMIR, 2021).

O aprendizado de máquina semi supervisionado trata os dados rotulados da mesma maneira que o aprendizado de máquina supervisionado e utiliza pontos de dados não rotulados para tornar o modelo mais coerente. Exemplos não rotulados baseiam-se no progresso realizado com dados rotulados (LUDERMIR, 2021).

3 Descrição dos Algoritmos Implementados

3.1 Árvore de Decisão

A Árvore de Decisão é um dos algoritmos mais conhecidos na área de ML, especialmente quando se trata de problemas de classificação. Sua popularidade advém de sua capacidade de lidar com conjuntos de dados complexos de forma relativamente simples, enquanto ainda oferece resultados robustos e interpretáveis (Bishop, 2006).

Uma característica distintiva da Árvore de Decisão é sua estrutura hierárquica em forma de árvore. Cada nó interno representa uma decisão baseada em uma característica dos dados, enquanto as folhas representam as classes ou categorias de saída. Esse formato espelha o processo humano de tomada de decisões, tornando a interpretação do modelo bastante intuitiva (Breiman et al., 1984).

O processo de construção de uma Árvore de Decisão envolve a divisão do conjunto de dados em subconjuntos menores com base em características específicas, de modo que a pureza das classes (ou homogeneidade) seja maximizada em cada subdivisão. Isso é feito de maneira iterativa, onde o algoritmo busca as melhores decisões em cada nó da árvore para otimizar a precisão das previsões (Hastie, Tibshirani, & Friedman, 2009).

Um dos principais benefícios da Árvore de Decisão é sua capacidade de lidar com dados numéricos e categóricos, além de lidar naturalmente com interações não lineares entre as características. Isso a torna uma escolha versátil para uma ampla gama de problemas de classificação, desde diagnósticos médicos até análise de crédito e detecção de fraudes (Ludermir, 2021).

Outro ponto forte é a capacidade de interpretar e explicar as decisões do modelo. Por exemplo, ao analisar uma árvore gerada, é possível identificar quais características são mais importantes para a classificação, permitindo insights valiosos para especialistas de domínio (Braga & Prates, 2020).

No entanto, é importante ressaltar que Árvores de Decisão podem ser suscetíveis a overfitting (sobreajuste) se não forem adequadamente controladas durante o processo de construção. Estratégias como poda da árvore, limitação da profundidade e uso de critérios de parada são comuns para mitigar esse problema e garantir a generalização do modelo (Mahesh, 2020).

Árvore de Decisão representa não apenas uma ferramenta poderosa para classificação de dados, mas também uma abordagem intuitiva e interpretável que continua a desempenhar um papel central no arsenal de técnicas de Machine Learning para resolver uma variedade de problemas do mundo real.

3.2 Support Vector Machines (SVM) (Máquina de Vetores de Suporte)

O Support Vector Machine (SVM), ou Máquina de Vetores de Suporte, é um dos algoritmos mais eficientes para problemas de classificação. Sua abordagem baseia-se na busca por um hiperplano de separação ótimo entre as diferentes classes, maximizando a margem entre os pontos mais próximos de diferentes classes, conhecidos como vetores de suporte (Bishop, 2006).

A essência do SVM reside na capacidade de encontrar não apenas um hiperplano de separação entre as classes, mas o melhor hiperplano possível, garantindo uma separação clara e robusta mesmo em conjuntos de dados complexos e com sobreposição entre as classes. Isso é crucial para a precisão das previsões, especialmente em problemas de classificação binária onde a distinção entre duas classes é de interesse primordial (Hastie, Tibshirani, & Friedman, 2009).

A principal intuição por trás do SVM é criar um "corredor" ou "margem" entre as diferentes classes, onde a largura dessa margem é maximizada. Isso é alcançado encontrando o hiperplano que separa as classes e mantém a maior distância possível dos pontos mais próximos de cada classe. Esses pontos mais próximos são os vetores de suporte, e o hiperplano é determinado por eles (Fontana, 2020).

Uma das grandes vantagens do SVM é sua capacidade de lidar bem com conjuntos de dados de alta dimensionalidade e complexidade. Ele é especialmente

útil em problemas onde existem muitas características (features) e é necessário um bom desempenho de generalização. Além disso, SVMs podem ser estendidos para problemas de classificação multiclasse por meio de estratégias como One-vs-All (um contra todos) ou One-vs-One (um contra um) (Braga & Prates, 2020).

Entretanto, é importante destacar que a eficácia do SVM depende da escolha adequada do kernel, que define a função de transformação dos dados para um espaço de maior dimensionalidade onde a separação linear pode ser mais clara. Kernels comuns incluem o linear, polinomial e RBF (*Radial Basis Function*), cada um com suas características e impacto na capacidade de generalização do modelo (Ludermir, 2021).

SVM é uma ferramenta robusta para problemas de classificação que exige uma separação clara entre classes, maximizando a margem entre elas. Sua eficiência em lidar com conjuntos de dados complexos e a capacidade de generalização o tornam uma escolha valiosa em uma variedade de aplicações, desde diagnósticos médicos até detecção de fraudes e reconhecimento de padrões (Mahesh, 2020).

3.3 Redes Neurais Artificiais (ANN)

As Redes Neurais Artificiais (ANNs) são inspiradas no funcionamento do cérebro humano. Assim como o cérebro humano é composto por neurônios interconectados, as ANNs são estruturas compostas por camadas de neurônios artificiais que trabalham em conjunto para aprender padrões complexos nos dados (Bishop, 2006).

O cerne da eficácia das Redes Neurais Artificiais reside na sua capacidade de aprender de forma não linear e capturar relações intrincadas nos dados. Isso as torna extremamente versáteis e aplicáveis a uma ampla variedade de tarefas de classificação, desde reconhecimento de padrões em imagens até diagnósticos médicos e análise de texto (Hastie, Tibshirani, & Friedman, 2009).

A estrutura básica de uma ANN consiste em camadas de neurônios, geralmente divididas em camada de entrada, camadas ocultas (*hidden layers*) e camada de saída. Cada neurônio em uma camada está conectado aos neurônios da camada seguinte por meio de conexões ponderadas, que são ajustadas durante o processo de treinamento para otimizar o desempenho da rede (Fontana, 2020).

Durante o treinamento, as ANNs utilizam algoritmos de aprendizado, como o backpropagation (retropropagação), para ajustar os pesos das conexões de forma a minimizar uma função de custo, que representa a diferença entre as previsões da rede e os valores reais dos dados de treinamento. Esse processo iterativo permite que a rede neural aprenda a partir dos exemplos apresentados e melhore sua capacidade de generalização para dados não vistos (Braga & Prates, 2020).

Uma das principais vantagens das Redes Neurais Artificiais é sua capacidade de lidar com grandes volumes de dados e aprender representações hierárquicas e abstratas dos mesmos. Isso as torna particularmente eficazes em problemas onde as relações entre as características dos dados são complexas e não lineares (Ludermir, 2021).

No entanto, as ANNs também apresentam desafios, como a necessidade de um conjunto de dados de treinamento representativo e suficiente, além da escolha adequada da arquitetura da rede, número de camadas, neurônios por camada e algoritmos de otimização (Mahesh, 2020).

3.4 Árvore de Regressão

A Árvore de Decisão, conhecida por sua eficácia em problemas de classificação, também se destaca em problemas de regressão, onde o objetivo principal é prever um valor numérico a partir de um conjunto de características (features). Esta versatilidade torna a Árvore de Decisão uma excelente escolha para uma ampla gama de aplicações que envolvem previsão de valores contínuos (Bishop, 2006).

Ao contrário da abordagem de classificação, onde as folhas da árvore representam classes ou categorias de saída, na regressão, cada folha da árvore representa um valor de saída específico. Durante o processo de construção da árvore de decisão para regressão, o algoritmo busca dividir os dados de forma a minimizar a variação dos valores previstos dentro de cada nó folha (Breiman et al., 1984).

A estrutura em forma de árvore permite capturar relações não lineares entre as características de entrada e a variável de saída, o que é particularmente útil em problemas onde os padrões não são facilmente modelados por métodos lineares. Além disso, a interpretabilidade da árvore de decisão na regressão é uma vantagem significativa, pois permite compreender visualmente como as diferentes características influenciam a previsão final (Hastie, Tibshirani, & Friedman, 2009).

A construção de uma árvore de decisão para regressão envolve o processo de divisão dos dados em subconjuntos com base em determinadas características, de modo que a variância dos valores de saída seja minimizada em cada nó folha. Essa divisão é realizada de forma recursiva até que um critério de parada seja atendido, como profundidade máxima da árvore ou número mínimo de amostras por nós (Fontana, 2020).

É importante mencionar que, assim como em problemas de classificação, as árvores de decisão na regressão também podem ser suscetíveis a overfitting. Estratégias como a poda da árvore, limitação da profundidade e ajuste dos parâmetros são comuns para evitar o sobreajuste e garantir a generalização do modelo para novos dados (Ludermir, 2021).

3.5 Redes Neurais Artificiais (ANN) em Problemas de Regressão

As Redes Neurais Artificiais (ANNs) também podem ser aplicadas em problemas de regressão. Ao contrário de abordagens lineares mais tradicionais, as ANNs têm a capacidade de aprender relações complexas e não lineares entre as variáveis de entrada e saída, tornando-as especialmente úteis em contextos onde os

padrões de dados são intrincados e não podem ser capturados por métodos mais simples (Bishop, 2006).

O processo de regressão com Redes Neurais Artificiais envolve a construção de uma arquitetura de rede neural composta por camadas de neurônios interconectados. Cada neurônio recebe um conjunto de entradas, realiza um cálculo ponderado dessas entradas e aplica uma função de ativação para produzir uma saída. Durante o treinamento, as conexões entre os neurônios são ajustadas iterativamente para minimizar a diferença entre as previsões da rede e os valores reais dos dados de treinamento (Hastie, Tibshirani, & Friedman, 2009).

Uma das principais vantagens das ANNs na regressão é sua capacidade de lidar com problemas não lineares e capturar padrões complexos nos dados. Isso significa que a rede pode aprender a partir de relações não lineares entre as variáveis de entrada e saída, extrapolando para além das capacidades de modelos lineares mais simples (Fontana, 2020). Como resultado, as ANNs são adequadas para uma ampla gama de aplicações, incluindo previsão de séries temporais, modelagem de fenômenos naturais complexos e análise de dados multidimensionais (Ludermir, 2021).

A adaptabilidade das ANNs também é uma característica marcante. Diferentes arquiteturas de redes neurais, como redes feedforward, redes recorrentes (RNNs) e redes neurais convolucionais (CNNs), podem ser aplicadas de acordo com a natureza dos dados e as nuances do problema em questão. Isso permite uma flexibilidade significativa na modelagem e otimização do desempenho da rede para alcançar resultados precisos e confiáveis na tarefa de regressão (Braga & Prates, 2020).

3.6 Stochastic Gradient Descent (SGD)

O *Stochastic Gradient Descent* (SGD) ou Gradiente Descendente Estocástico é um dos algoritmos mais utilizados para otimização em aprendizado de máquina,

especialmente em redes neurais. Sua popularidade se deve à simplicidade de implementação e eficiência computacional, tornando-o adequado para grandes conjuntos de dados. Em essência, o SGD é uma variação do *Gradient Descent* (GD), um método iterativo que ajusta os parâmetros de um modelo para minimizar uma função de custo, que mede a diferença entre as previsões do modelo e os valores reais (Pedregosa et al., 2011).

O algoritmo de gradiente descendente tradicional calcula o gradiente da função de custo em relação a todos os exemplos do conjunto de treinamento, o que pode ser computacionalmente caro e impraticável para grandes volumes de dados. O SGD resolve esse problema calculando o gradiente com base em um único exemplo de treinamento (ou um pequeno lote de exemplos) por vez, o que reduz significativamente o custo computacional e permite atualizações mais frequentes dos parâmetros. Esta abordagem não só torna o processo mais eficiente, mas também permite que o modelo aprenda e se adapte mais rapidamente (Bishop, 2006).

O SGD apresenta várias vantagens e desvantagens. Entre as vantagens, destacam-se a eficiência computacional e a convergência rápida. O cálculo do gradiente baseado em um único exemplo torna o SGD muito mais rápido e eficiente em termos de memória, especialmente para grandes conjuntos de dados. Além disso, em muitos casos, o SGD converge mais rapidamente do que o GD tradicional, pois realiza atualizações frequentes dos parâmetros. Outra vantagem importante é a generalização do modelo. As flutuações introduzidas pelas atualizações baseadas em exemplos individuais podem ajudar a escapar de mínimos locais, contribuindo para uma melhor generalização do modelo (Hastie, Tibshirani, & Friedman, 2009).

Por outro lado, o SGD também tem suas desvantagens. Devido ao uso de um único exemplo por iteração, as atualizações podem ser muito ruidosas, o que pode levar a uma trajetória de convergência instável. A escolha da taxa de aprendizado é crítica; uma taxa muito alta pode fazer o algoritmo divergir, enquanto uma taxa muito baixa pode resultar em convergência lenta. Para mitigar o ruído, uma variação popular do SGD é o Mini-batch SGD, que usa um pequeno lote de exemplos em vez de um único exemplo para cada atualização. Isso equilibra a eficiência computacional e a estabilidade da convergência (Pedregosa et al., 2011).

O SGD é amplamente utilizado em várias áreas de aprendizado de máquina, incluindo o treinamento de redes neurais, sistemas de recomendação e processamento de linguagem natural (NLP). No treinamento de redes neurais profundas, o SGD é o método mais utilizado para a otimização devido à sua eficiência. Em sistemas de recomendação, o SGD é usado para ajustar modelos de recomendação em larga escala (Bishop, 2006).

3.7 K-means

A clusterização K-means é uma das técnicas mais amplamente utilizadas em aprendizado de máquina e análise de dados para agrupar um conjunto de objetos de modo que os objetos dentro do mesmo grupo (ou cluster) sejam mais semelhantes entre si do que aos objetos em outros grupos. O K-means é especialmente popular devido à sua simplicidade e eficiência computacional (Mahesh, 2020).

O K-means parte de uma ideia simples: dividir um conjunto de dados em K clusters, onde K é um número definido previamente pelo usuário. O algoritmo segue os seguintes passos básicos. Primeiro, escolhe-se K pontos iniciais chamados de centróides. Estes pontos podem ser escolhidos aleatoriamente a partir do conjunto de dados ou através de algum método heurístico. Em seguida, cada ponto do conjunto de dados é atribuído ao centróide mais próximo, formando assim K clusters iniciais. A proximidade é geralmente medida usando a distância euclidiana. Depois, calcula-se a média dos pontos em cada cluster para encontrar o novo centróide. Esta média representa o novo "centro" do cluster. Os passos de atribuição e atualização dos centróides são repetidos até que os centróides não mudem significativamente, ou seja, até que a posição dos centróides estabilize. O algoritmo é garantido para convergir, embora possa não encontrar a solução globalmente ótima e dependa da inicialização dos centróides (Bishop, 2006).

O K-means oferece várias vantagens. É relativamente rápido e pode ser aplicado a grandes conjuntos de dados. O algoritmo é simples de entender e implementar, e funciona bem com grandes volumes de dados, tornando-se escalável

para muitas aplicações práticas (Mahesh, 2020). No entanto, também apresenta algumas desvantagens. O número de clusters K deve ser especificado previamente, o que pode ser desafiador sem conhecimento prévio dos dados. O K-means assume que os clusters têm formas esféricas e tamanhos similares, o que pode não ser adequado para todos os tipos de dados. Pontos de dados atípicos podem influenciar significativamente os centróides e, consequentemente, a formação dos clusters. Além disso, diferentes escolhas iniciais dos centróides podem levar a diferentes soluções de clusters (Hastie, Tibshirani, & Friedman, 2009).

Existem algumas variações e melhorias do algoritmo K-means. Uma delas é o K-means++, um método de inicialização que seleciona centróides iniciais de maneira menos aleatória, o que pode melhorar a convergência e a qualidade dos clusters. Outra variação é o Mini-batch K-means, que usa mini-lotes de dados para atualizar os centróides, tornando o algoritmo mais eficiente para grandes conjuntos de dados (Bishop, 2006).

O K-means é amplamente utilizado em diversas áreas. Em segmentação de mercado, agrupa clientes com base em características semelhantes para campanhas de marketing direcionadas. Em compressão de imagens, reduz o número de cores em uma imagem agrupando pixels com cores semelhantes. E no agrupamento de documentos, agrupa documentos com base em similaridades de conteúdo para facilitar a recuperação de informações (Mahesh, 2020).

3.8 DBSCAN

O DBSCAN é uma técnica de clusterização baseada em densidade que se diferencia de abordagens tradicionais, como o K-means, por sua capacidade de identificar clusters de forma arbitrária e lidar efetivamente com ruído nos dados. Em vez de exigir que o número de clusters seja especificado a priori, o DBSCAN depende de dois parâmetros principais: o raio de vizinhança e o número mínimo de pontos necessários para formar um cluster (Bishop, 2006).

A abordagem do DBSCAN é intuitiva e robusta. Começa-se selecionando um ponto aleatório no conjunto de dados e verificando se há pelo menos pontos dentro de um raio ao redor desse ponto. Se a condição for atendida, um novo cluster é iniciado e o algoritmo se expande para incluir todos os pontos densamente conectados. Este processo continua até que todos os pontos densamente conectados sejam agrupados, formando clusters de diferentes formas e tamanhos. Pontos que não se encaixam em nenhum cluster denso são considerados ruídos, o que permite ao DBSCAN lidar efetivamente com outliers e regiões de baixa densidade nos dados (Mahesh, 2020).

Uma das principais vantagens do DBSCAN é sua capacidade de detectar clusters de forma arbitrária e sua robustez em relação à escolha dos parâmetros. Ao contrário do K-means, que assume que os clusters têm formas esféricas e tamanhos similares, o DBSCAN é mais flexível e pode identificar clusters de qualquer forma e tamanho. Além disso, o DBSCAN é menos sensível à inicialização e à escolha dos parâmetros, tornando-o uma escolha atraente para conjuntos de dados complexos e de alta dimensionalidade (Hastie, Tibshirani, & Friedman, 2009).

4 Descrição da Interface Gráfica para Interação com os Algoritmos de Aprendizado de Máquina

Interface gráfica do usuário (em inglês *Graphic User Interface*, também conhecida pela sigla GUI) é um método para facilitar a interação do usuário com o computador por meio da utilização de um conjunto de imagens e objetos pictóricos (ícones, janelas) além de texto. (LUDERMIR, 2021).

Neste capítulo é descrito as principais características da GUI desenvolvida neste TCC para o uso de alguns algoritmos de aprendizado de máquina.

4.1 Entrada de dados

A entrada de dados é realizada exclusivamente através de arquivos Excel. Esse formato de entrada foi escolhido devido à sua popularidade e facilidade de uso para manipulação de grandes conjuntos de dados. Além disso, para garantir a precisão e a eficiência no processamento dos dados, apenas colunas com valores numéricos são consideradas para o treinamento dos algoritmos. Para a extração dos arquivos em formato CSV, foi empregada a biblioteca Pandas, um pacote Python de código aberto que facilita o uso de dados relacionais ou rotulados, permitindo a análise e manipulação (NumFOCUS, Inc., 2023).

4.2 Tela em React

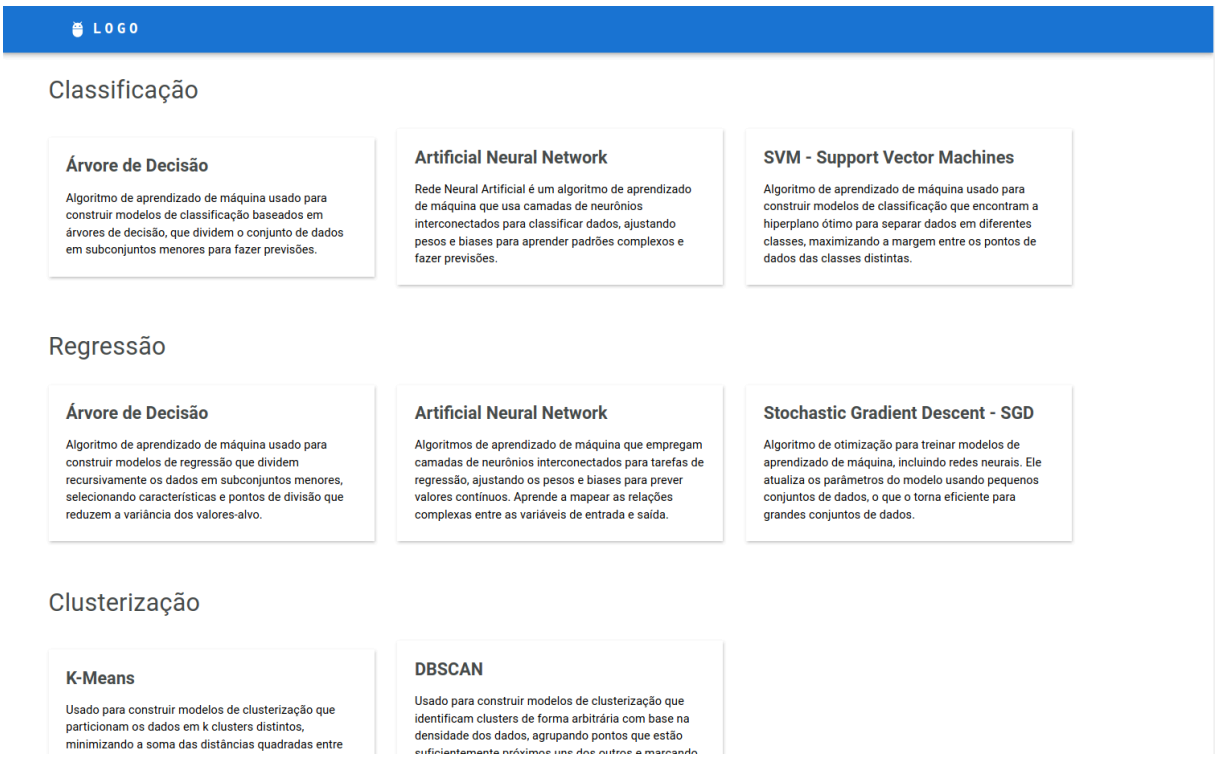
O projeto desenvolvido utiliza a biblioteca React, uma biblioteca JavaScript de código aberto mantida pelo Facebook e por uma comunidade de desenvolvedores individuais e empresas. React é amplamente utilizada para a construção de interfaces de usuário, permitindo a criação de componentes reutilizáveis que facilitam a manutenção e escalabilidade de grandes aplicações web (Facebook, Inc., 2023).

Primeira Tela: Listagem de Algoritmos

Na primeira tela da aplicação, como pode ser visto na Figura 1, é listado os algoritmos disponíveis, organizados por área. Cada algoritmo é representado por um *card* que exibe informações básicas sobre ele. A separação por área facilita a navegação e a busca pelos algoritmos específicos de interesse do usuário.

Ao clicar em um dos cards, o usuário é redirecionado para uma segunda página, onde é possível utilizar o algoritmo selecionado.

Figura 1 - Listagem de algoritmos



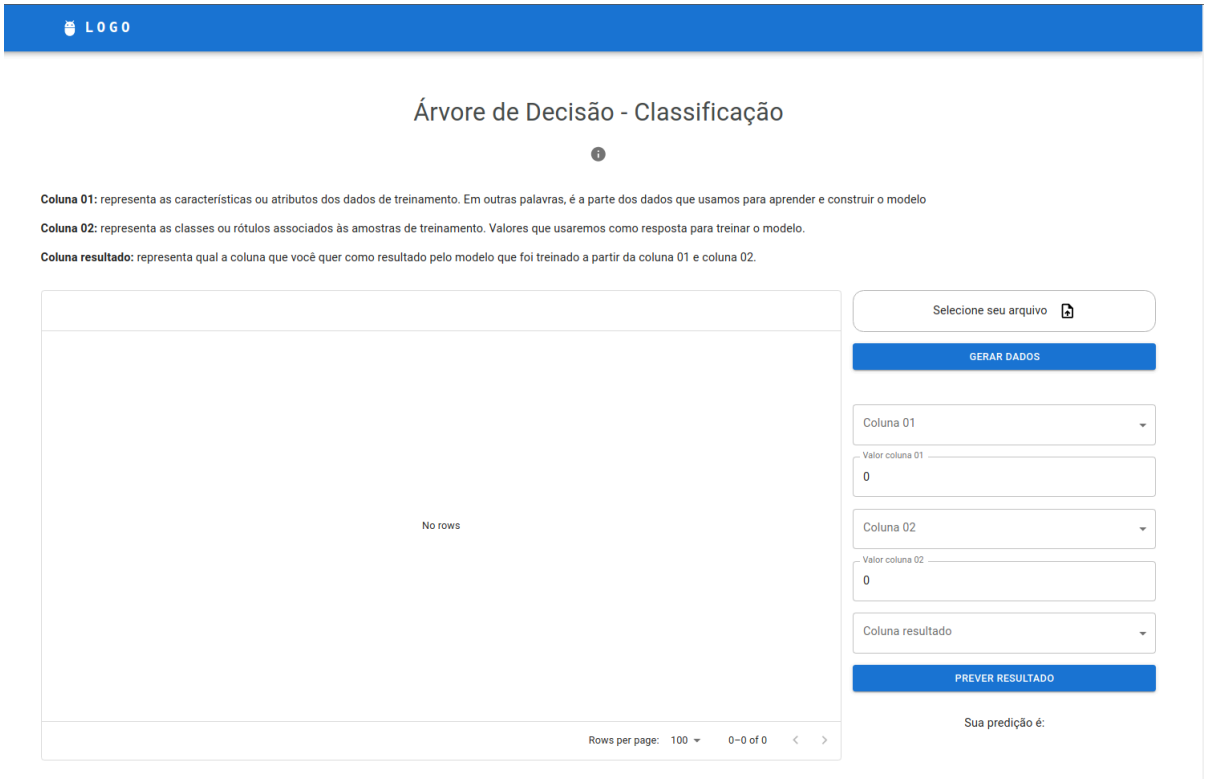
Fonte - Elaborado pelo autor

Segunda Tela: Utilização do Algoritmo

Na segunda página, o usuário encontra uma interface detalhada para o uso do algoritmo escolhido. A interface é composta pelos seguintes elementos:

1. Explicação dos Parâmetros: na parte superior da página, há uma explicação detalhada dos parâmetros que serão utilizados pela API para o treinamento do algoritmo, que pode ser visto na parte superior da Figura 2. Esses parâmetros são essenciais para configurar corretamente o algoritmo de acordo com os dados fornecidos.

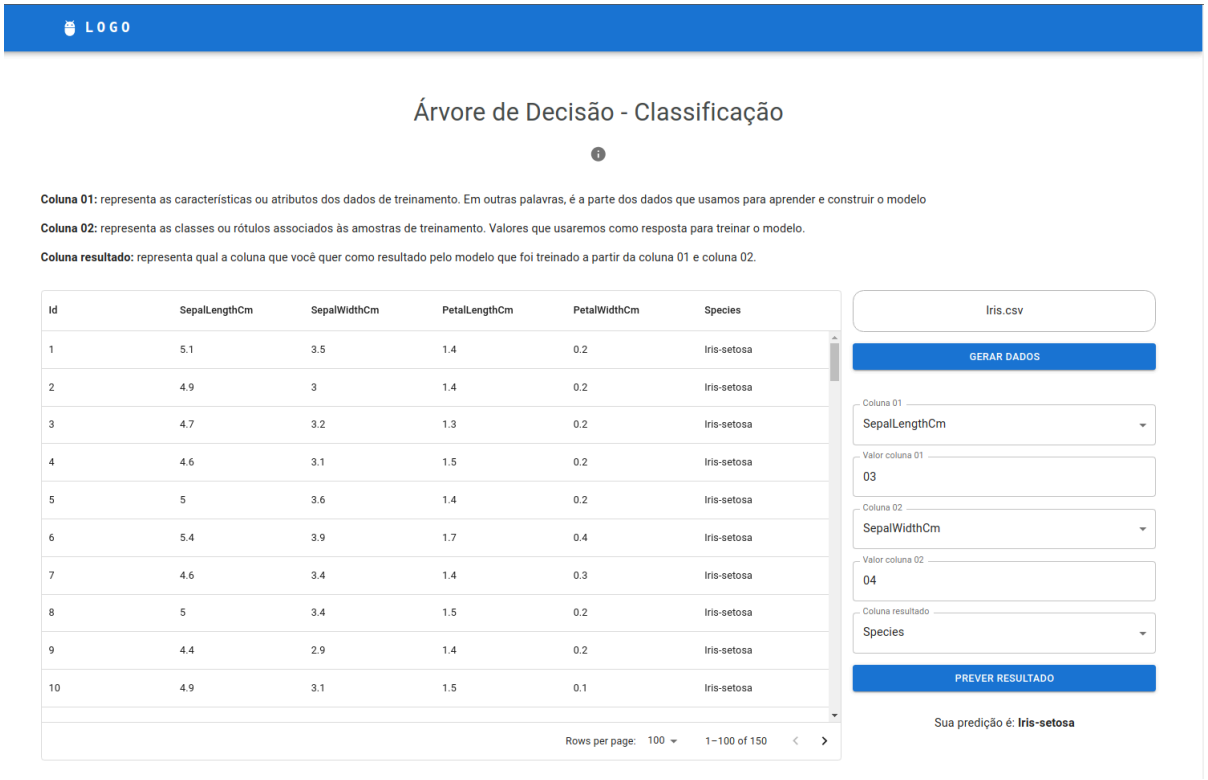
Figura 2 - Página do algoritmo



Fonte - Elaborado pelo autor

2. Grid de Dados: abaixo da explicação dos parâmetros, um grid exibe os dados que são lidos pela API, que pode ser visualizado na Figura 3. Esses dados são fornecidos através da primeira parte do formulário localizado à direita da tela.

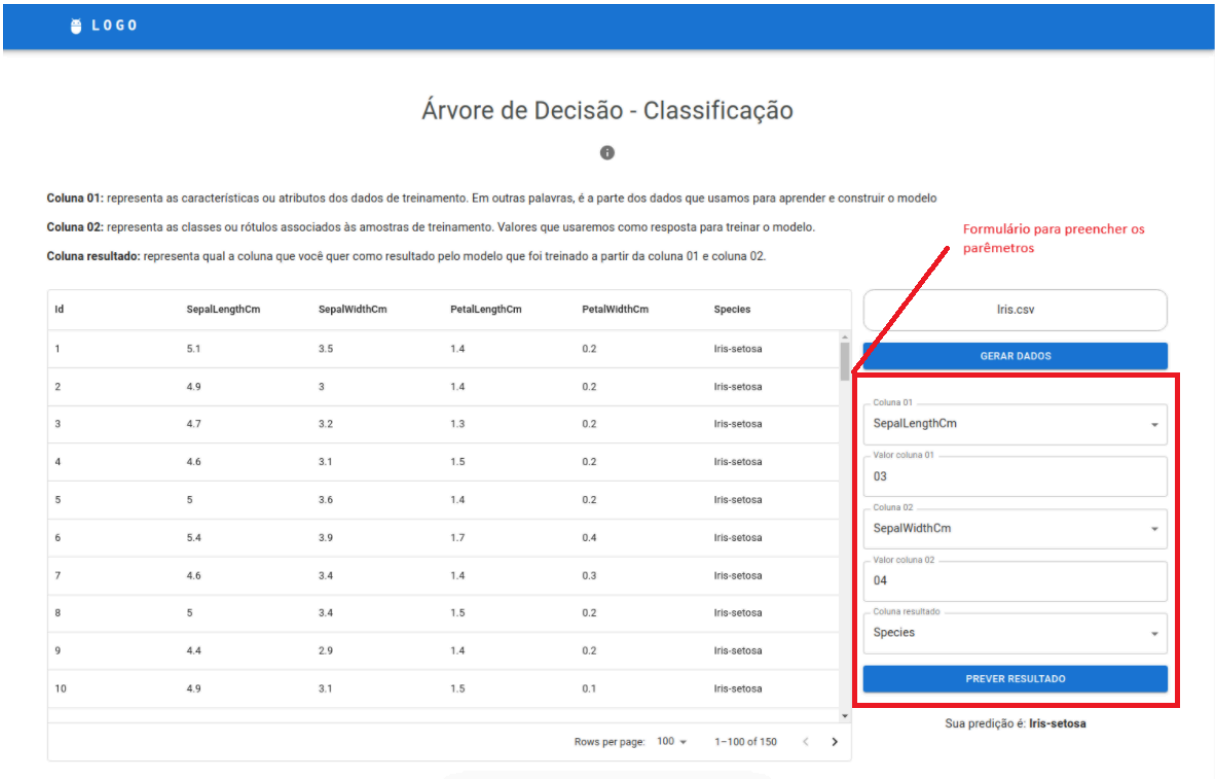
Figura 3 - Upload de arquivo



Fonte - Elaborado pelo autor

3. Campos do Formulário: logo abaixo do grid de dados, estão os campos que foram descritos anteriormente. Estes campos permitem ao usuário inserir e ajustar os valores dos parâmetros necessários para o treinamento do algoritmo. Pode-se perceber destacado em vermelho na Figura 4.

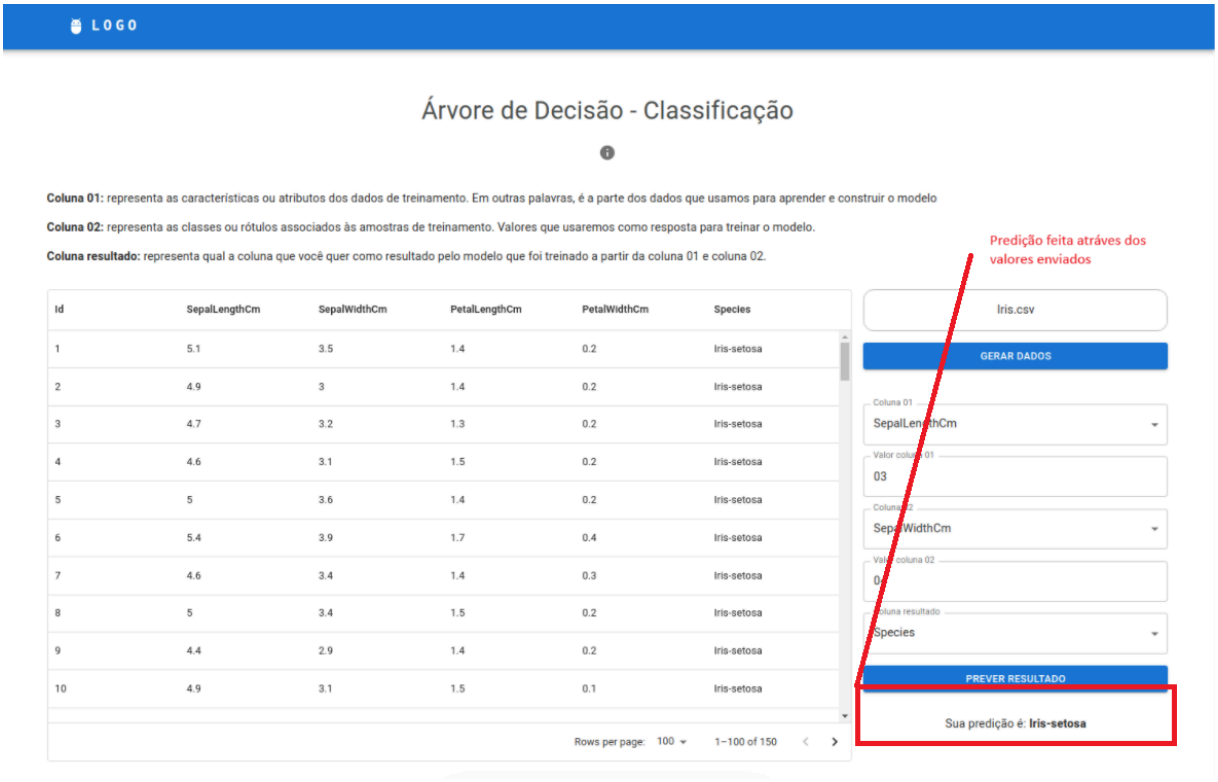
Figura 4 - Formulário



Fonte - Elaborado pelo autor

4. Predição do algoritmo: após o preenchimento dos parâmetros é possível fazer a predição segundo os *inputs* do formulário. A predição pode ser vista na Figura 5 abaixo.

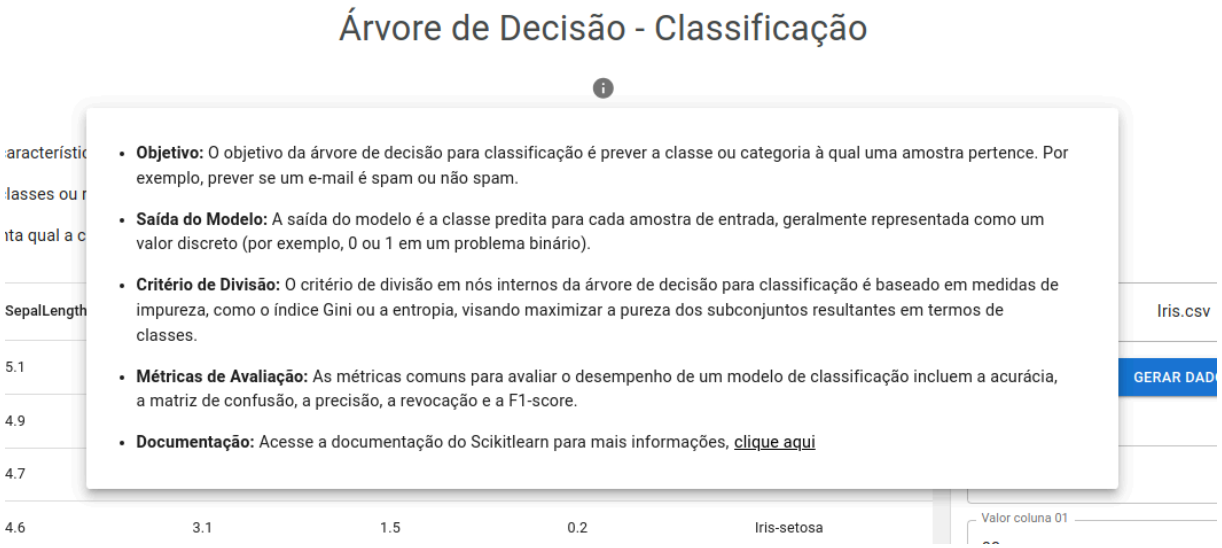
Figura 5 - Predição do algoritmo



Fonte - Elaborado pelo autor

5. Botão de Visualização Detalhada: no início da página, há um botão que, ao ser clicado, exibe uma descrição detalhada com informações sobre os objetivos do algoritmo, a saída esperada do modelo, critérios de divisão dos dados, métricas de avaliação e um link para a documentação oficial da biblioteca utilizada, o Scikit-learn (Pedregosa et al., 2011). Essa descrição é apresentada pela Figura 6.

Figura 6 - Descrição detalhada do algoritmo



Fonte - Elaborado pelo autor

4.3 Desenvolvimento da API

A API do projeto foi construída utilizando FastAPI, um framework web moderno e de código aberto para a construção de APIs em Python. FastAPI é conhecido por sua alta performance, facilitando a criação de APIs rápidas e robustas, e oferecendo suporte completo a anotações de tipo do Python, o que melhora a auto completação e a verificação de tipos em tempo de desenvolvimento.

4.3.1 Arquitetura e Funcionalidades

A API foi projetada utilizando a arquitetura MVC (*Model-View-Controller*), uma abordagem que separa a lógica de negócios, a interface do usuário e o controle de entrada, facilitando a manutenção e escalabilidade do código. A arquitetura MVC permite uma organização clara do código em três componentes principais:

1. Model (Modelo): responsável pela lógica de dados da aplicação. Ele define as estruturas de dados e a lógica para armazenar e recuperar dados.

2. View (Visão): responsável pela apresentação dos dados. Ele define a estrutura de exibição dos dados ao usuário.

3. Controller (Controlador): responsável pelo controle de fluxo e lógica de aplicação. Ele manipula a entrada do usuário, interage com o modelo e seleciona a visão apropriada para a resposta.

A API contém uma rota específica para cada algoritmo disponível, além de rotas auxiliares, como a rota de upload de arquivos. Este design modular permite que a API seja facilmente extensível, adicionando novas rotas para futuros algoritmos ou funcionalidades.

4.3.2 Fluxo de Funcionamento

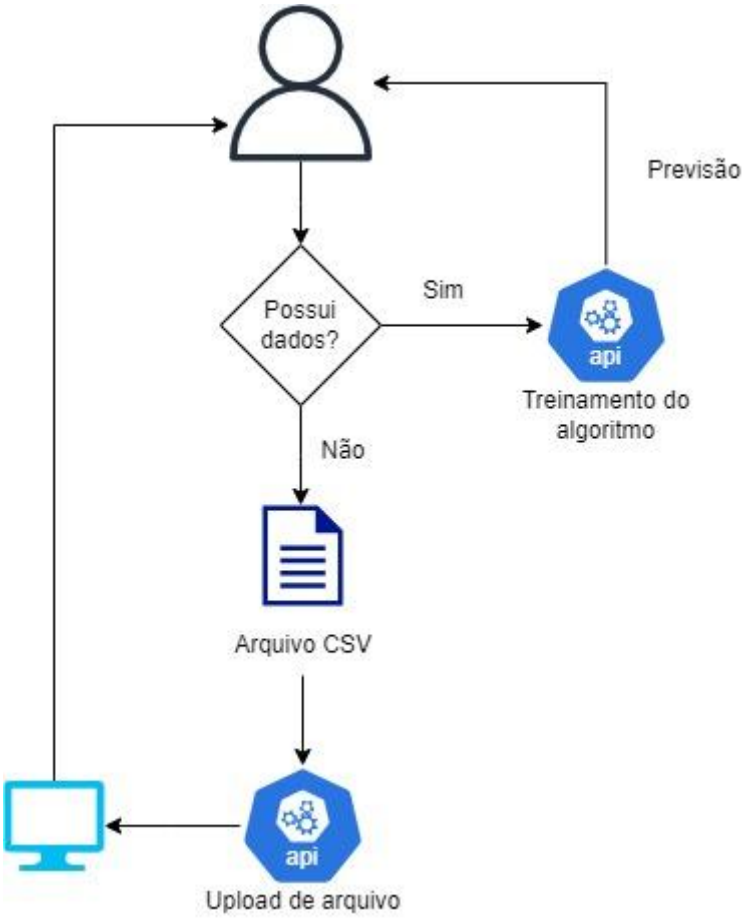
- Upload de Arquivo: o usuário inicia o processo fazendo o upload de um arquivo através de uma rota específica da API. Esse arquivo geralmente contém os dados que serão utilizados para treinar o algoritmo.

- Envio de Payload com Parâmetros: após o upload do arquivo, o usuário pode enviar um payload contendo os parâmetros pré-definidos necessários para o treinamento do algoritmo. Esses parâmetros incluem detalhes como o tipo de algoritmo a ser utilizado, parâmetros para o treinamento, e outras configurações específicas.

- Treinamento do Algoritmo: com os dados e parâmetros recebidos, a API inicia o treinamento do algoritmo selecionado. O treinamento é realizado utilizando os dados fornecidos e o ajuste do modelo conforme os parâmetros especificados.

- Retorno da Previsão: após o treinamento, a API retorna a previsão ou o resultado do modelo treinado. Essa resposta pode incluir métricas de desempenho, previsões específicas para novos dados, ou outros outputs relevantes definidos pelo usuário. Esse fluxo é ilustrado na Figura 7.

Figura 7 - Fluxo dos algoritmos



Fonte - Elaborado pelo autor

Em resumo, todos os algoritmos implementados seguem esse fluxo estruturado, diferenciando-se apenas nos parâmetros específicos enviados para a API, o que permite uma abordagem sistemática e adaptável para a aplicação de diferentes técnicas de aprendizado de máquina.

5 CONSIDERAÇÕES FINAIS E PROPOSTAS PARA TRABALHOS FUTUROS

Neste Trabalho, foi apresentado um sistema web com o objetivo de facilitar o uso de algoritmos de aprendizado de máquina, utilizando uma interface intuitiva e diversas ferramentas. O objetivo proposto foi alcançado, proporcionando uma plataforma eficaz para o aprendizado de conceitos complexos de machine learning. A plataforma desenvolvida visa democratizar o acesso às técnicas de aprendizado de máquina, permitindo que usuários, mesmo sem profundo conhecimento técnico, possam explorar e aplicar esses algoritmos em diferentes contextos.

O sistema desenvolvido apresenta uma interface gráfica acessada via WEB que possui as informações necessárias para o uso de alguns algoritmos de aprendizado de máquina. A GUI interage com a biblioteca de ML Scikit-Learning, que efetivamente implementa os algoritmos de ML utilizados. Através da interface, o usuário pode enviar os dados para o treinamento e definir um conjunto de parâmetros, que variam dependendo do algoritmo escolhido.

Além disso, é importante destacar que, embora o sistema tenha sido desenvolvido e testado internamente, não foram realizados testes com o público-alvo. Isso significa que a avaliação da usabilidade e da eficácia do sistema em contextos reais de uso ainda não foi completamente explorada.

Para melhorias em trabalhos futuros, várias possibilidades foram identificadas. Uma das principais sugestões é a alteração dos algoritmos já desenvolvidos, permitindo a adição de novos parâmetros e a incorporação de novos formatos de dados para o treinamento. Além disso, o desenvolvimento e a implementação de novos algoritmos, disponibilizados pela biblioteca Scikit-learn, também se destacam como uma área promissora para a expansão.

No que diz respeito à interface do sistema, é sugerido que sejam disponibilizados mais meios de visualização para avaliar a efetividade do treinamento, como gráficos e imagens detalhadas. Essas ferramentas adicionais de visualização podem oferecer insights mais profundos e facilitar a compreensão dos resultados obtidos.

Por fim, é recomendada a melhoria contínua da interface aplicando outras teorias de UI/UX. Isso pode incluir a adoção de práticas modernas de design de interface, garantindo que o sistema não só seja funcional, mas também ofereça uma experiência de usuário agradável e intuitiva.

Essas melhorias propostas visam não apenas aperfeiçoar a funcionalidade do sistema, mas também tornar a aprendizagem de máquinas mais acessível e compreensível para os usuários, contribuindo assim para um ensino mais eficaz e dinâmico na área de aprendizado de máquinas.

Em conclusão, o sistema desenvolvido neste trabalho oferece uma ferramenta versátil para o ensino de aprendizado de máquinas, com um potencial significativo para evoluir e se adaptar às necessidades futuras da área.

REFERÊNCIAS

BISHOP, Christopher M. Pattern Recognition and Machine Learning. Secaucus, NJ: Springer, 2006.

BRAGA, Eduardo Pacheco Carreiro; PRATES, Matheus Henrique do Amaral. Desenvolvimento de um framework de redes neurais com interface gráfica. 2020. Trabalho de conclusão de curso (Bacharelado em engenharia elétrica) - Universidade Tecnológica Federal do Paraná, [S. l.], 2020.

FONTANA, Éliton. Introdução aos Algoritmos de Aprendizagem Supervisionada, 2020. Disponível em: https://fontana.paginas.ufsc.br/files/2018/03/apostila_ML_pt2.pdf. Acesso em: 12/04/2023.

HASTIE, Trevor; TIBSHIRANI, Robert; FRIEDMAN, Jerome. The Elements of Statistical Learning. 2009.

LUDERMIR, T. B. Inteligência artificial e aprendizado de máquina: estado atual e tendências. Estudos Avançados, SciELO Brasil, v. 35, p. 85–94, 2021.

MAHESH, B. Machine learning algorithms-a review. International Journal of Science and Research (IJSR).[Internet], v. 9, p. 381–386, 2020.

MAGER, Gabriela Botelho. Interface gráfica para aplicativo computacional: desenvolvimento de uma interface baseada em critérios de ergonomia, usabilidade e design. 2004. Trabalho de conclusão de curso (Programa de Pós-Graduação em Engenharia de Produção) - Universidade Federal de Santa Catarina, [S. l.], 2004

SANTOS, Fábio Oliveira do. Uso de algoritmos de aprendizado de máquina na resolução de falhas em redes móveis / Fábio Oliveira Dos santos; Natalia Castro Fernandes, orientadora. Niterói, 2019. 100 f.: il. Dissertação (mestrado)-Universidade Federal Fluminense, Niterói, 2019. Disponível em: http://dx.doi.org/10.22409/PPGEET.2019.m.07519655709. Acesso em: 20/03/2024.

SIQUEIRA, Guilherme; RODRIGUES, Gustavo; FEITOSA, Eduardo; KREUTZ, Diego. QuickAutoML: Uma ferramenta para treinamento automatizado de modelos de aprendizado de máquina. In: Escola regional de redes de computadores (ERRC), 19. 2021, Charqueadas/RS. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2021. p. 85-90. Disponível em: https://doi.org/10.5753/errc.2021.18547. Acesso em: 19/03/2024.

Breiman, L., Friedman, J., Stone, C., & Olshen, R. (1984). Classification and regression trees. CRC press.

NumFOCUS, Inc. Pandas, 2023. Package overview. Disponível em: https://pandas.pydata.org/docs/getting_started/overview.html. Acesso em: 05, jun. 2024.

Facebook, Inc. (2023). React – A JavaScript library for building user interfaces. Recuperado de https://reactjs.org/

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830.