MLCASE: Plataforma Automatizada de Desenvolvimento de Modelos de *Machine Learning*

Simplificando e acelerando o processo de construção de modelos para Cientistas de Dados

Mestrado em Ciência de Dados

Bruna Filipa Menó de Sousa

Leiria, Março de 2025

MLCASE: Plataforma Automatizada de Desenvolvimento de Modelos de *Machine Learning*

Simplificando e acelerando o processo de construção de modelos para Cientistas de Dados

Mestrado em Ciência de Dados

Bruna Filipa Menó de Sousa

Trabalho de Projeto realizado sob a orientação do Professor Doutor Ricardo Manuel da Silva Malheiro

Leiria, Março de 2025

Originalidade e Direitos de Autor

O presente Relatório de projeto é original, elaborado unicamente para este fim, tendo sido devidamente citados todos os autores cujos estudos e publicações contribuíram para o elaborar.

Agradecimentos

Quero dedicar este momento especial para expressar a minha profunda gratidão a todas as pessoas que tornaram possível a realização deste projeto. A entrega deste projeto marca o fim de uma importante fase da minha vida, que não teria sido possível sem a ajuda de várias pessoas.

Quero começar por expressar o meu agradecimento ao Professor Doutor Ricardo Malheiro, pela sua disponibilidade e orientação, que desempenharam um papel fundamental no desenvolvimento deste relatório de projeto.

Quero também agradecer a todos os professores da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria, que, de diferentes maneiras, partilharam os seus conhecimentos e, sem dúvida, enriqueceram a elaboração deste trabalho com valiosos *insights*.

À minha família, namorado e amigos, quero agradecer pelo seu inabalável apoio e incentivo. Sem a vossa compreensão, encorajamento e amor, esta jornada teria sido ainda mais desafiadora.

Aos meus colegas de curso, pelas longas horas de estudo e colaboração, agradeço por partilharem o entusiasmo e os desafios deste percurso. Juntos, crescemos e aprendemos de forma inestimável.

Este projeto de tese representa o culminar de esforço, dedicação, resiliência, aprendizagem e entusiasmo. A todos os que me ajudaram a superar as dificuldades que enfrentei ao longo deste caminho e que celebraram comigo as inúmeras alegrias que ele me proporcionou, o meu sincero obrigada.

Resumo

O desenvolvimento de modelos de *Machine Learning* é um processo intrinsecamente complexo, composto por múltiplas etapas, como o pré-processamento de dados, a seleção de algoritmos, a otimização de hiperparâmetros e a avaliação de desempenho. Este projeto propõe o desenvolvimento de uma plataforma automatizada, denominada MLCASE, com o objetivo de simplificar e acelerar estas etapas, permitindo que os cientistas de dados otimizem o seu tempo e concentrem os seus esforços na análise de resultados e na obtenção de *insights* estratégicos. A MLCASE foi concebida com base na *framework* Streamlit, proporcionando uma interface intuitiva e interativa, enquanto automatiza tarefas fundamentais como a seleção de *features*, a pesquisa e otimização de hiperparâmetros e a avaliação de modelos de aprendizagem automática. Adicionalmente, a plataforma integra a criação automática de relatórios técnicos e gráficos analíticos, promovendo a democratização do acesso a técnicas avançadas de *Machine Learning* e a sua aplicação eficiente em áreas como saúde, finanças, marketing e ciência social.

Palavras-chave: Aprendizagem Automática, AutoML, Desenvolvimento de Modelos, Automatização, Otimização de Hiperparâmetros, Streamlit.

Abstract

The development of Machine Learning models is inherently complex, encompassing multiple stages such as data preprocessing, algorithm selection, hyperparameter optimization, and performance evaluation. This project proposes the development of an automated platform named MLCASE, designed to simplify and accelerate these stages, enabling data scientists to optimize their time and focus on result analysis and strategic insights. MLCASE is built using the Streamlit framework, providing an intuitive and interactive interface while automating critical tasks such as *feature* selection, hyperparameter tuning, and machine learning model evaluation. Additionally, the platform includes automated creation of technical reports and analytical visualizations, promoting the democratization of access to advanced Machine Learning techniques and their efficient application in fields such as healthcare, finance, marketing, and social sciences.

Keywords: Machine Learning, AutoML, Automatization, Model Development, Hyperparameter Optimization, Streamlit.

Índice

Lista de Figuras

Lista de tabelas

Lista de siglas e acrónimos

A

B

C

D

E

F

G

I

J

K

L

M

P

R

S

T

1. Introdução

A crescente complexidade dos modelos de Aprendizagem Automática (*Machine Learning*, ML), aliada à vasta quantidade de dados atualmente disponíveis, tornou evidente a necessidade de ferramentas que tornem o processo de desenvolvimento de modelos mais ágil e acessível. A construção de modelos de ML envolve diversas etapas, como o pré-processamento de dados, a seleção de algoritmos e a otimização de hiperparâmetros, tarefas que exigem tempo considerável e um conhecimento técnico aprofundado. Para profissionais sem formação especializada ou que operam sob restrições temporais, estas exigências podem representar um desafio significativo.

Neste contexto, a automação do fluxo de trabalho de ML surge como uma solução promissora, permitindo reduzir a carga de trabalho manual associada ao desenvolvimento de modelos. As plataformas de AutoML (*Automated Machine Learning*) foram concebidas com esse propósito, possibilitando que cientistas de dados direcionem os seus esforços para a análise e interpretação dos resultados, em vez de se dedicarem a processos repetitivos e morosos. Apesar dos avanços nesta área, muitas das ferramentas e bibliotecas disponíveis para a automação de ML ainda apresentam limitações em termos de usabilidade, flexibilidade e personalização. Grande parte dessas soluções requer conhecimentos técnicos avançados para serem utilizadas de forma eficiente, restringindo a sua acessibilidade a um público mais amplo e limitando o seu impacto no processo de democratização da aprendizagem automática.

Nos últimos anos, novas tendências têm vindo a transformar o panorama do ML, nomeadamente a adoção de *Deep Learning* automatizado e a utilização de *Large Language Models* (LLMs) para tarefas de análise de dados e predição. Estas abordagens avançadas permitem a extração de padrões complexos a partir de grandes volumes de dados, tornando os modelos mais robustos e eficazes em diversas aplicações.

Plataformas como Google Vertex AI, H2O.ai e Azure ML destacam-se pela integração de ferramentas avançadas de automação, garantindo escalabilidade e compatibilidade com infraestruturas em nuvem. Estas soluções facilitam o desenvolvimento e a implementação de modelos de ML, reduzindo a necessidade de intervenção manual e

tornando o processo mais acessível a diferentes perfis de utilizadores (Hutter, Kotthoff, & Vanschoren, 2019).

Este projeto propõe o desenvolvimento de uma plataforma automatizada de ML, denominada MLCASE, com o objetivo de simplificar e acelerar o processo de criação de modelos de aprendizagem automática. Desenvolvida com o *framework* Streamlit, a plataforma será intuitiva e acessível, proporcionando aos utilizadores uma interface interativa que permitirá realizar tarefas essenciais — como pré-processamento de dados, treino de modelos, seleção de atributos, comparação e avaliação de modelos — sem a necessidade de programação manual.

A revisão da literatura desempenha um papel fundamental neste projeto, fornecendo um suporte teórico sólido para a análise das ferramentas de automação existentes em ML e justificando a necessidade do desenvolvimento de uma nova solução. A crescente complexidade dos modelos de aprendizagem automática reforça a importância da automação, tornando essencial a implementação de estratégias eficazes para otimizar o desenvolvimento e a utilização destes modelos (Xin He, 2021).

A presente revisão tem um duplo propósito. Primeiramente, pretende oferecer uma análise abrangente das abordagens, técnicas e ferramentas atualmente disponíveis para a automação das diferentes etapas do desenvolvimento de modelos de ML. A implementação eficiente de processos automatizados é essencial para responder à crescente exigência por análises de dados complexas e modelos preditivos de elevada precisão. Além disso, a revisão visa identificar lacunas e desafios nas soluções existentes, de forma a justificar o desenvolvimento de uma nova plataforma. Compreender as limitações das ferramentas de AutoML disponíveis permite definir com maior precisão as funcionalidades e inovações necessárias para tornar a nova solução mais eficiente e acessível. Através desta análise crítica, torna-se possível avaliar as soluções atuais e compreender as suas vantagens e limitações, orientando assim o desenvolvimento de uma plataforma que responda de forma mais eficaz às necessidades do setor.

O ML tem vindo a tornar-se uma componente essencial nas soluções de análise de dados em diversos domínios, como saúde, finanças, marketing e ciências sociais. No entanto, o desenvolvimento de modelos de ML pode ser um processo altamente complexo, especialmente quando envolve grandes volumes de dados, exigindo recursos

computacionais significativos e um investimento substancial de tempo. Dada esta realidade, a automação assume um papel cada vez mais relevante, permitindo que profissionais de ciência de dados e áreas afins otimizem os seus fluxos de trabalho e se concentrem na análise e interpretação de resultados estratégicos.

Este projeto pretende contribuir para a área da automação em ML, através do desenvolvimento de uma ferramenta versátil, acessível e eficiente, desenhada para responder às principais necessidades dos profissionais da área. A plataforma MLCASE, desenvolvida com a *framework* Streamlit[1], visa proporcionar uma automação integrada e personalizável, facilitando todo o ciclo de desenvolvimento de modelos, desde o pré-processamento de dados até à avaliação dos modelos gerados.

A revisão da literatura constitui o suporte teórico para este desenvolvimento, permitindo identificar as melhores abordagens e definir o caminho mais adequado para a construção da plataforma. Ao integrar contributos de autores proeminentes na área, o objetivo não é apenas colmatar as lacunas existentes, mas também disponibilizar uma solução robusta e eficaz, que melhore a acessibilidade e eficiência da automação no desenvolvimento de modelos de ML.

Este relatório encontra-se estruturado em quatro capítulos principais. No **Capítulo 2**, é apresentada uma revisão do estado da arte, abordando os conceitos fundamentais de *Machine Learning* e uma análise comparativa das principais ferramentas de AutoML disponíveis. O **Capítulo 3** descreve a metodologia adotada no desenvolvimento da plataforma MLCASE, detalhando as tecnologias, *frameworks* e estratégias utilizadas na sua implementação. No **Capítulo 4**, são apresentados os resultados obtidos, acompanhados de uma análise crítica e de uma discussão sobre o desempenho e aplicabilidade da plataforma. Por fim, no **Capítulo 5**, são expostas as conclusões do projeto, destacando as principais contribuições da plataforma MLCASE para a área de automação em *Machine Learning* e apontando possíveis direções futuras para a sua evolução e expansão.

[1] *Framework* de código aberto que converte *scripts* Python em aplicações web interativas, com ênfase na simplicidade e rapidez de desenvolvimento (Snowflake Inc., 2025).

2. Estado da Arte

Este capítulo apresenta uma revisão da literatura relevante para este projeto, abordando conceitos fundamentais e as ferramentas existentes na área da aprendizagem automática (ML). O objetivo deste capítulo é compreender as principais abordagens utilizadas na automatização do desenvolvimento de modelos de ML e analisar as vantagens e limitações das soluções disponíveis. Serão analisadas e comparadas as ferramentas de AutoML, bem como as duas principais abordagens de aprendizagem automática: Supervisionada e Não-Supervisionada, amplamente utilizadas no desenvolvimento de modelos. Adicionalmente, serão descritas as principais etapas do fluxo de trabalho de ML, incluindo os processos realizados antes da implementação dos algoritmos e as métricas utilizadas para avaliar o desempenho dos modelos.

A **secção 2.1** introduz os conceitos fundamentais de *Machine Learning*, incluindo os diferentes tipos de aprendizagem e os desafios inerentes à implementação de modelos. A **secção 2.2** apresenta a metodologia CRISP-DM, explicando as suas seis fases e a sua relevância na estruturação de projetos de ML.

Na **secção 2.3**, são analisadas e comparadas diversas ferramentas de AutoML, incluindo soluções baseadas em código, interfaces gráficas e ferramentas de uso académico e empresarial. A **secção 2.4** explora os algoritmos de aprendizagem supervisionada, destacando as respetivas vantagens e desvantagens, enquanto a **secção 2.5** foca-se na aprendizagem não supervisionada, apresentando os principais algoritmos e as suas características. A **secção 2.6** discute o pré-processamento de dados, abordando técnicas como limpeza, normalização e codificação de variáveis.

A **secção 2.7** apresenta o método *Grid Search* para a otimização de hiperparâmetros, comparando-o com abordagens alternativas como *Random Search* e Otimização Bayesiana. A **secção 2.8** explica a importância da divisão dos dados em treino, validação e teste, destacando a técnica de validação cruzada. A **secção 2.9** descreve os principais modelos de ML, incluindo classificação, regressão, *clustering* e modelos de associação. A **secção 2.10** aprofunda as métricas de avaliação de modelos, enquanto a **secção 2.11** apresenta a tecnologia Streamlit justificando a escolha para o desenvolvimento da plataforma MLCASE.

2.1. Conceitos Básicos de *Machine Learning*

O ML é um ramo da Inteligência Artificial (IA) dedicado ao desenvolvimento de algoritmos que analisam dados, identificam padrões e fazem previsões de forma autónoma, sem necessidade de programação explícita. A capacidade dos modelos de ML de identificar padrões e *insights* úteis revolucionou áreas como saúde, finanças, marketing e várias outras indústrias (Buskirk, 2018). O objetivo central do ML é permitir que máquinas ou sistemas extraiam conhecimento a partir de grandes volumes de dados e façam previsões ou categorizem dados de forma automática.

A área de ML integra conceitos de várias disciplinas, como IA estatística, otimização, ciência cognitiva e teoria da informação. Este caráter interdisciplinar torna o campo vasto e adaptável a múltiplos domínios científicos e práticos. Com a evolução da tecnologia e o aumento exponencial da disponibilidade de dados, a procura por soluções automatizadas que otimizem o desenvolvimento de modelos de ML tornou-se urgente.

Os modelos de ML dividem-se em três categorias principais, conforme representado na Figura 1: Aprendizagem Supervisionada, Aprendizagem Não-Supervisionada e Aprendizagem Por Reforço (Bishop, 2006). Apesar do seu potencial, a implementação de modelos de ML exige um elevado nível de conhecimento técnico. As etapas envolvidas, desde o tratamento e preparação de dados até à seleção de algoritmos e ajuste de hiperparâmetros, podem ser morosas e complexas. Assim, surge a necessidade de automatizar estes processos para democratizar o acesso ao ML e aumentar a eficiência das análises.

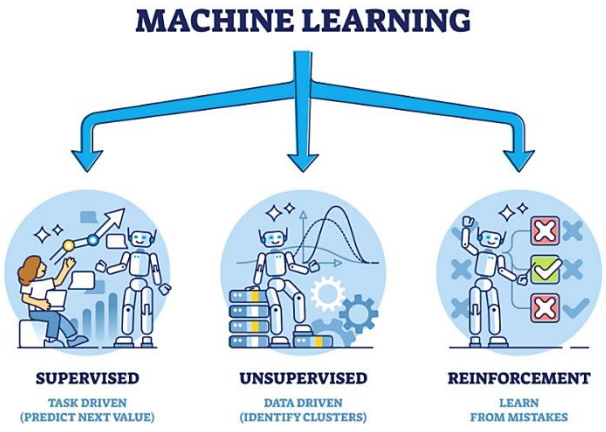


Figura 1. Tipos de *Machine Learning*

Fonte: Concannon, 2024

Segundo Xin He, Kaiyong Zhao e Xiaowen Chu (Xin He, 2021) o conceito de AutoML surgiu como uma solução para simplificar o desenvolvimento de modelos preditivos, automatizando etapas críticas do fluxo de trabalho, como o pré-processamento de dados, a seleção de algoritmos e a otimização de hiperparâmetros. As ferramentas de AutoML têm como objetivo reduzir a necessidade de conhecimento especializado em ML, tornando a aprendizagem automática mais acessível a um público mais amplo, ao mesmo tempo que aumentam a eficiência de analistas e especialistas de domínio.

Desafios na Implementação de ML

Apesar das vantagens associadas ao uso de AutoML, as ferramentas disponíveis atualmente enfrentam desafios significativos, que devem ser abordados para garantir maior eficácia e adoção generalizada. Dentre esses desafios, destacam-se:

- **Explicabilidade e Interpretabilidade:**

Muitos modelos gerados automaticamente são frequentemente descritos como "*Black-Box"* (Caixa-Preta), o que dificulta a interpretação e explicação dos resultados obtidos. Para mitigar essa limitação, têm sido aplicadas técnicas como SHAP (*Shapley Additive Explanations*) e LIME (*Local Interpretable Model-agnostic Explanations*), que visam aumentar a transparência e permitir a interpretação das decisões tomadas pelos modelos. O SHAP é baseado na teoria de valores de *Shapley*, proveniente da teoria dos jogos, e atribui uma contribuição individual a cada característica (*feature*) para a previsão feita por um modelo. Essa técnica cria gráficos que mostram como cada característica impacta o resultado do modelo, permitindo uma explicação detalhada e interpretável (Lundberg & Lee, 2017). Por outro lado, o LIME gera explicações locais ao criar modelos lineares simples ao redor de cada ponto de dado a ser interpretado. Esse método permite aproximar o comportamento do modelo em pequenas regiões, proporcionando *insights* rápidos sobre a lógica subjacente às previsões (Ribeiro, Singh, & Guestrin, 2016).

No entanto, para além dessas abordagens explicativas aplicadas a modelos de caixa-preta, existem modelos que são inerentemente interpretáveis, como as árvores de decisão e os modelos baseados em regras. As árvores de decisão são modelos que particionam os dados em regras lógicas claras, facilitando a compreensão do processo de tomada de decisão (Razzano, Brandi, Piscitelli, & Capozzoli, 2025). De forma semelhante, os modelos baseados em regras seguem uma estrutura explícita, permitindo uma explicação

mais transparente e compreensível das decisões, sendo amplamente utilizados em contextos onde a interpretabilidade é essencial (Kopanja, Savic, & Longo, 2025). Estes modelos fornecem transparência desde a sua conceção, contrastando com técnicas como SHAP e LIME, que são usadas para explicar modelos mais complexos e menos interpretáveis.

- **Escalabilidade e Tempo de Execução:**

O ajuste de hiperparâmetros e a otimização de fluxos de trabalhos podem ser computacionalmente exigentes, especialmente ao lidar com grandes volumes de dados. Este fator pode afetar negativamente o desempenho e a viabilidade prática das ferramentas de AutoML em ambientes com recursos computacionais limitados. A elevada exigência computacional deve-se à necessidade de explorar várias combinações de hiperparâmetros e configurações de fluxos de trabalho, frequentemente utilizando abordagens como procura em grelha (*Grid Search*) e otimização bayesiana (*Bayesian Optimization*). Embora estas técnicas possam ser aceleradas através do uso de processamento paralelo e computação em nuvem, continuam a representar um desafio para infraestruturas mais limitadas (Hutter, Kotthoff, & Vanschoren, 2019).

- **Personalização e Flexibilidade**:

Embora ferramentas como *Auto-sklearn*[2] e TPOT[3] ofereçam recursos avançados para personalização de fluxos de trabalho e ajuste de modelos, essas soluções ainda exigem conhecimentos técnicos especializados para personalizações mais específicas, tornando-as menos acessíveis a utilizadores iniciantes. Em particular, a criação e a personalização de fluxos de trabalho exigem familiaridade com programação e conceitos avançados de ML, como seleção de atributos (*features*), codificação de variáveis categóricas e validação cruzada. Além disso, essas ferramentas nem sempre fornecem interfaces gráficas intuitivas, o que dificulta o acesso para utilizadores sem experiência prévia. Para ultrapassar estas limitações, é necessário investir na criação de interfaces mais acessíveis e guias interativos que simplifiquem o processo de configuração de fluxos de trabalho personalizados (Feurer, et al., 2015) (Olson, Bartley, Urbanowicz, & Moore, 2016).

[2] Ferramenta de AutoML e um substituto direto para um estimador do *scikit-learn* (Freiburg, 2022)
[3] Ferramenta de AutoML em Python que otimiza pipelines de ML utilizando programação genética (S.Olson, 2024).

Por exemplo, em casos de deteção de fraudes ou análise de séries temporais, os utilizadores podem precisar de configurar manualmente algoritmos específicos ou criar combinações personalizadas de fluxos de trabalho, o que exige conhecimentos mais aprofundados. Ferramentas como o *RapidMiner*[4] tentam resolver essa limitação ao combinar interfaces gráficas com suporte para código personalizado, permitindo flexibilidade para utilizadores avançados e simplicidade para iniciantes (Altair Engineering, 2024). Tendências recentes apontam para o crescimento de plataformas *no-code e low-code*, que reduzem a necessidade de programação manual, democratizando o acesso à personalização.

Investir na criação de interfaces mais acessíveis e guias interativos para simplificar a configuração de fluxos de trabalho personalizados é uma área promissora. Isso pode melhorar a adoção de ferramentas de AutoML em ambientes acadêmicos e empresariais, tornando a tecnologia mais inclusiva e eficaz.

2.2.Metodologia CRISP-DM

A metodologia CRISP-DM (*Cross-Industry Standard Process for Data Mining*) é um modelo independente de tecnologia, amplamente utilizado para orientar o desenvolvimento de projetos de *Machine Learning* e *Data Science*. O seu principal objetivo é tornar os processos de mineração de dados mais fáceis, rápidos e repetíveis, definindo um conjunto de atividades fundamentais para a implementação de um projeto analítico (Mariscal, Marban, & Fernandez, 2010).

O CRISP-DM estrutura o ciclo de vida de um projeto de *Data Science* em seis fases principais, conforme representado na Figura 2. Essas fases não precisam ser estritamente sequenciais, permitindo a repetição e o refinamento contínuo de etapas conforme necessário (Azevedo & Santos, 2008).

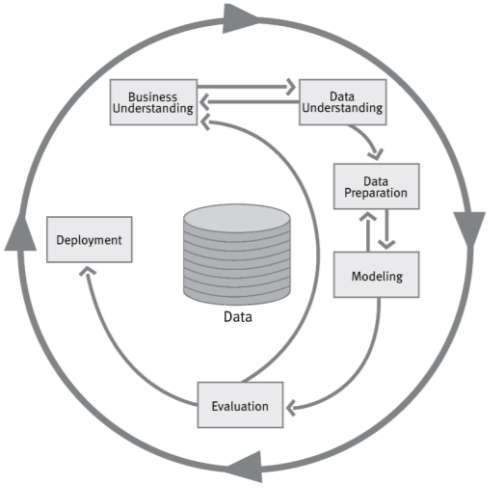


Figura 2. Fases do CRISP-DM

Fonte: Shimaoka, Ferreira, & Goldman, 2024

[4] Interface gráficas intuitiva para a conceção de processos de análise, eliminando a necessidade de programação manual, mas também permitindo a execução em modo *batch* (Hofmann & Klinkenberg, 2014).

Fases do CRISP-DM (Shimaoka, Ferreira, & Goldman, 2024)

- **Compreensão do Negócio (*Business Understanding*)** – Esta fase centra-se na compreensão dos objetivos e requisitos do projeto do ponto de vista do negócio. O cientista de dados traduz essas necessidades numa definição de problema em *Machine Learning* e elabora um plano de projeto adequado.

- **Compreensão dos Dados (*Data Understanding*)** – Fase dedicada à exploração inicial dos dados, permitindo que o cientista de dados compreenda melhor a estrutura, qualidade e possíveis padrões presentes. Nesta etapa, são formuladas hipóteses iniciais que guiam as fases subsequentes.

- **Preparação dos Dados (*Data Preparation*)** – O foco desta etapa é a construção de um conjunto de dados adequado para a modelação, através de tarefas como seleção, transformação e limpeza dos dados. Dependendo da complexidade do projeto, esta fase pode ser repetida várias vezes até que os dados estejam corretamente preparados.

- **Modelação (*Modeling*)** – Envolve a seleção e aplicação de técnicas de *Machine Learning*, ajustando parâmetros e testando diferentes abordagens para resolver o problema proposto. Durante esta fase, podem ser detetadas inconsistências nos dados, levando a ajustes nas fases anteriores.

- **Avaliação (*Evaluation*)** – Após a construção dos modelos, é necessário validar os resultados e garantir que atendem aos objetivos do projeto e cumprem os critérios de desempenho estabelecidos. Para isso, são utilizadas métricas de avaliação, e é feita uma revisão detalhada das fases anteriores para garantir a qualidade do modelo final.

- **Implementação (*Deployment*)** – Nesta fase, o modelo é integrado num ambiente produtivo, podendo assumir diferentes formatos, desde um relatório técnico até uma solução automatizada para toda a organização (por exemplo, um sistema de recomendação para um *e-commerce*).

2.3. Comparação de Ferramentas de AutoML

A evolução do ML e a crescente necessidade de automação no desenvolvimento de modelos preditivos impulsionaram a criação de diversas ferramentas de AutoML. Cada uma destas ferramentas foi desenvolvida para abordar desafios específicos, como a simplificação de tarefas repetitivas, a otimização de hiperparâmetros e a criação de fluxos de trabalho eficientes. No entanto, estas soluções apresentam diferenças significativas em termos de funcionalidades, flexibilidade e aplicabilidade.

2.3.1. Ferramentas Baseadas em Código e Algoritmos Avançados

Entre as ferramentas mais técnicas e flexíveis encontram-se o *Auto-sklearn* e o TPOT. Ambas foram projetadas para utilizadores com experiência em programação e estatística, oferecendo suporte avançado para otimização de modelos e automação de fluxos de trabalho.

O *Auto-sklearn* é amplamente utilizado devido à sua integração com a biblioteca *Scikit-learn*[5] e à utilização de otimização bayesiana para ajustar modelos. Esta abordagem permite encontrar soluções robustas para problemas de classificação e regressão. Contudo, a necessidade de conhecimentos técnicos avançados torna-o pouco acessível para utilizadores iniciantes e não especialistas (Feurer, et al., 2015).

Por outro lado, o TPOT destaca-se pelo uso de algoritmos genéticos, que permitem a exploração automática de combinações complexas de fluxos de trabalho (Olson, Bartley, Urbanowicz, & Moore, 2016). Esta abordagem torna-o altamente adaptável e flexível, mas, devido ao elevado consumo de recursos computacionais, pode ser inadequado para grandes conjuntos de dados ou análises que exijam resultados rápidos.

2.3.2. Ferramentas Baseadas em Interfaces Gráficas Intuitivas

Em contraste com as soluções baseadas em código, ferramentas como Google AutoML (Google, 2024) e H2O.aia (Enterprise, 2024) foram desenvolvidas para utilizadores

[5] Biblioteca de código aberto para ML que suporta aprendizagem supervisionada e não supervisionada, oferecendo ferramentas para ajuste de modelos, pré-processamento de dados, seleção e avaliação de modelos (Scikit-learn Developers, 2025).

menos experientes, oferecendo interfaces gráficas simplificadas e integração com serviços na nuvem.

O Google AutoML permite treinar e avaliar modelos sem necessidade de programação, sendo particularmente eficaz em tarefas específicas, como visão computacional e processamento de linguagem natural. No entanto, apresenta custos elevados e pouca transparência em relação aos algoritmos subjacentes, dificultando a interpretação dos resultados (Google Cloud, 2024).

O H2O.ai, por sua vez, suporta algoritmos avançados, como redes neuronais, e integra-se com plataformas em nuvem. Destaca-se pela escalabilidade e pela capacidade de lidar com grandes volumes de dados, tornando-o adequado para análises corporativas. Contudo, a personalização limitada e a exigência de infraestrutura robusta podem restringir a sua aplicação em ambientes académicos ou em pequenas empresas (Enterprise, 2024).

2.3.3. Ferramentas Académicas e Acessíveis

A ferramenta MLJAR *Studio* oferece uma interface simplificada e funcionalidades automatizadas para a criação e análise de modelos de ML, incluindo relatórios detalhados. É uma solução acessível, especialmente adequada para pequenas empresas, investigadores e utilizadores com menos experiência técnica. No entanto, apresenta limitações em termos de memória e processamento, o que pode torná-la menos adequada para análises de grandes volumes de dados ou aplicações empresariais complexas (MLJAR, s.d.).

2.3.4. Ferramentas Empresariais e na Nuvem

Ferramentas como *Azure Machine Learning* e *Amazon SageMaker Autopilot* foram projetadas para empresas e utilizadores avançados que necessitam de soluções escaláveis e altamente personalizáveis.

O *Azure Machine Learning* é um serviço na nuvem que acelera e gere o ciclo de vida dos projetos de ML, permitindo a criação, treino, implementação e monitorização de modelos.

Suporta *frameworks* como PyTorch[6], TensorFlow[7] e *scikit-learn*, oferecendo ferramentas para MLOps[8], automatização de fluxos de trabalho e integração segura em aplicações empresariais com controlo de acesso baseado em funções (Microsoft, 2024).

O *Amazon SageMaker Autopilot* destaca-se pela documentação detalhada e integração com a AWS, tornando-o ideal para empresas já inseridas nesse ecossistema. No entanto, o custo associado ao uso da infraestrutura da AWS pode ser uma barreira para projetos com recursos limitados (Amazon SageMaker, 2024).

WEKA.io destaca-se como uma solução moderna de armazenamento distribuído e gestão de dados, projetada para cargas de trabalho intensivas em ML e AI. Com suporte nativo para *frameworks* como TensorFlow e PyTorch, oferece alta performance e escalabilidade horizontal, sendo ideal para empresas que lidam com grandes volumes de dados e necessitam de integração contínua com plataformas de análise preditiva. Apesar das suas vantagens em termos de escalabilidade e desempenho, a WEKA.io apresenta custos elevados e requer configuração avançada para ambientes híbridos e *multicloud*, tornando-se mais adequada para empresas de grande porte e centros de investigação (WekaIO, 2024).

2.3.5. Ferramentas com foco na gestão de dados e fluxos de trabalho

Para ambientes que necessitam de gestão avançada de dados e *workflows* reprodutíveis, ferramentas como *RapidMiner* AutoModel (AltairEngineering, 2025) e Pachyderm (Pachyderm, 2023) destacam-se. Estas ferramentas combinam automação de ML com gestão de dados e versionamento, permitindo fluxos de trabalho reprodutíveis e escaláveis. Estas soluções são particularmente úteis para organizações que exigem controlo rigoroso sobre dados e modelos. Estas ferramentas destacam a importância de integrar *Data Engineering* e ML em plataformas unificadas, facilitando a implementação e monitorização de soluções em ambientes corporativos e académicos.

[6] Biblioteca de *deep learning* de código aberto, projetada para facilitar a construção e modificação de redes neurais e modelos de *Machine Learning* através de um gráfico de computação dinâmica (GeeksforGeeks, PyTorch Tutorial - Learn PyTorch with Examples, 2024).
[7] Biblioteca de código aberto desenvolvida pelo Google em 2015, amplamente utilizada para ML, computação numérica e *deep Learning* (Tech, 2024).
[8] Aplicação de práticas de DevOps (Desenvolvimento e Operações) em ML, automatizando o desenvolvimento, implementação e monitorização contínua de modelos (Microsoft, 2024).

O RapidMiner AutoModel facilita a integração com bases de dados empresariais e permite análises visuais intuitivas. No entanto, apresenta limitações em tarefas avançadas de personalização de modelos (AltairEngineering, 2025).

O Pachyderm é ideal para projetos que exigem escalabilidade e versionamento de dados, oferecendo integração com bibliotecas populares, como TensorFlow e *Scikit-learn*. Porém, requer configuração avançada e infraestrutura dedicada, o que pode limitar a sua adoção em ambientes menos preparados tecnologicamente (Pachyderm, 2023).

2.3.6. Estudos Comparativos

A análise comparativa evidencia um panorama diversificado de ferramentas de AutoML, adaptadas a diferentes contextos e necessidades. Ferramentas como *Auto-sklearn* e TPOT são ideais para utilizadores avançados que valorizam flexibilidade e controlo, enquanto soluções como Google AutoML e H2O.ai oferecem interfaces intuitivas para utilizadores menos experientes, embora apresentem limitações na personalização e custos elevados (Feurer, et al., 2015). Ferramentas empresariais como *Azure Machine Learning* e WEKA.io atendem a necessidades mais exigentes em termos de escalabilidade e integração com ambientes na nuvem (Microsoft, 2024). A introdução de plataformas como WEKA.io destaca a importância de soluções específicas para gestão de dados em larga escala, mas também reforça a necessidade de ferramentas híbridas que combinem usabilidade, personalização e eficiência (The AI-Native WEKA Data Platform, 2024).

A seleção da ferramenta de AutoML mais adequada depende de fatores como experiência do utilizador, complexidade do projeto e disponibilidade de recursos. Ferramentas como Google AutoML e BigML são indicadas para utilizadores iniciantes e projetos rápidos, enquanto soluções como *Auto-sklearn* e TPOT oferecem maior flexibilidade para cientistas de dados avançados. Por outro lado, ferramentas empresariais como H2O.ai, *Azure Machine Learning* e *Amazon SageMaker* destacam-se pela escalabilidade e integração com serviços na nuvem, sendo mais adequadas para grandes organizações.

A análise comparativa evidencia a importância de equilibrar custo, personalização e eficiência. Apesar das funcionalidades oferecidas pelas ferramentas de AutoML revisadas, ainda existem desafios importantes a serem superados. Embora estas ferramentas forneçam métricas e gráficos básicos, persiste uma lacuna significativa na

criação de relatórios compreensíveis para públicos técnicos e não técnicos, dificultando a interpretação dos resultados por *stakeholders*.

Neste contexto, a proposta MLCASE surge como uma alternativa em constante evolução, oferecendo um ambiente integrado e acessível para a automatização de ML, com o objetivo de preencher algumas das lacunas existentes no mercado. A sua principal inovação está na facilidade de utilização, tornando o processo de ML mais intuitivo e acessível a utilizadores com diferentes níveis de experiência. Desenvolvida com o *framework* Streamlit, a plataforma MLCASE combina simplicidade, eficiência e personalização, permitindo automatizar tarefas como pré-processamento de dados, treino de modelos, otimização de hiperparâmetros e avaliação de resultados de forma mais acessível e sem necessidade de conhecimentos avançados em programação. O seu principal objetivo é ser uma plataforma *user-friendly*, diferenciando-se por proporcionar uma experiência simplificada e interativa na criação de modelos preditivos. No entanto, a plataforma continua em desenvolvimento e há sempre melhorias a fazer. Além disso, dada a rápida e contínua evolução da área, serão necessários estudos e atualizações regulares para garantir que a MLCASE se mantém alinhada com os mais recentes avanços e melhores práticas em *Machine Learning.*

A Tabela 1 apresenta uma análise comparativa entre as principais ferramentas de AutoML, destacando as suas funcionalidades, limitações e aplicações. Essa análise permite avaliar as opções disponíveis no mercado e escolher a ferramenta mais adequada às necessidades específicas do projeto.

Tabela 1. Análise Comparativa das Ferramentas de AutoML: Funcionalidades, Limitações e Aplicações Práticas

Ferramenta	Vantagens	Desvantagens	Aplicações Práticas
Auto-sklearn	Utiliza otimização bayesiana; eficaz para classificação e regressão.	Pouco intuitivo para utilizadores não técnicos.	Problemas estruturados de regressão e classificação.
TPOT	Algoritmos genéticos para otimizar fluxos de trabalho; flexível e adaptável.	Exige recursos computacionais elevados.	Modelos complexos e experimentação rápida em ciência de dados.
H2O.ai	Suporte para grandes volumes de dados e integração com nuvem.	Limitações na personalização e infraestrutura avançada.	Grandes volumes de dados; análises corporativas escaláveis.
Google AutoML	Interface simplificada e fácil de usar; boa para iniciantes.	Custos elevados e falta de transparência nos algoritmos.	Classificação de imagens, NLP e análise de texto.
WEKA.io	Desempenho otimizado para grandes volumes de dados e fluxos de trabalho de ML.	Custo elevado e configuração complexa para ambientes híbridos.	Processamento de dados intensivos e fluxos de trabalho híbridos.
MLJAR AutoML	Simples e interpretável; gera relatórios detalhados.	Pode ser limitado para grandes volumes de dados.	Pequenas empresas e investigadores com poucos recursos.
Azure *Machine Learning*	Plataforma escalável e colaborativa na nuvem; integração com ferramentas populares.	Requer conhecimentos avançados; custos elevados.	Implementações colaborativas e personalizadas na nuvem.
Amazon SageMaker Autopilot	Automação completa na nuvem; *notebooks* detalhados com documentação.	Dependente do ecossistema AWS; custos proporcionais ao uso.	Projetos baseados na AWS; modelos escaláveis e documentados.
BigML	Interface gráfica para construção de modelos sem código.	Suporte limitado para modelos avançados, como redes neuronais.	Ferramentas de análise rápida para pequenos projetos.
Google Vertex AI	Flexibilidade avançada e integração com TensorFlow e PyTorch.	Custos elevados e dependência do ecossistema Google *Cloud*.	Modelos avançados baseados em TensorFlow e IA generativa.
RapidMiner AutoModel	Fluxos de trabalho automatizados e integração com bases de dados.	Personalização limitada para tarefas avançadas.	Integrações rápidas com bases de dados e análises visuais.
Pachyderm	Fluxo de trabalho escalável e reprodutível com gestão avançada de dados.	Configuração avançada e exigência de infraestrutura específica.	Projetos complexos com gestão rigorosa de dados e *workflows*.

2.4. Aprendizagem Supervisionada

A Aprendizagem Supervisionada é uma técnica amplamente utilizada para construir modelos preditivos em diferentes domínios. Consiste no treino de algoritmos com dados rotulados, nos quais cada instância de entrada possui uma correspondente saída conhecida, conforme representado na Figura 3. O modelo é treinado para reconhecer padrões que relacionam os inputs com os outputs, permitindo, posteriormente, fazer previsões em novos dados não rotulados (Géron, 2017).

Este tipo de aprendizagem é particularmente útil quando o objetivo é prever valores contínuos (regressão) ou categóricos (classificação). A aprendizagem supervisionada envolve um ciclo de treino, onde o modelo ajusta os seus parâmetros para minimizar o erro, seguido de uma fase de avaliação, onde o modelo é testado em novos dados para medir sua a precisão. O desempenho do modelo é geralmente calculado em termos de métricas como *Accuracy*, *Precision*, *Recall* e *F1-score* (Classificação), entre outras.

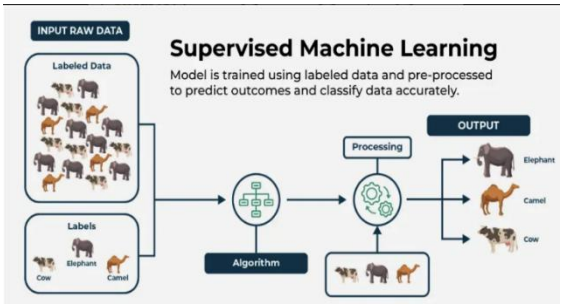


Figura 3. Aprendizagem Supervisionada

Fonte: Geeks for Geeks, 2024

2.4.1. Tipos de algoritmos de aprendizagem supervisionada

A aprendizagem supervisionada pode ser dividida em duas categorias principais: Regressão e Classificação. Cada uma destas categorias utiliza diferentes abordagens e algoritmos.

- **Regressão:** Utilizada para prever valores contínuos. O objetivo da regressão é modelar a relação entre uma ou mais variáveis independentes e uma variável dependente, geralmente contínua, como o preço de uma casa ou a previsão de temperatura.

- **Classificação:** Usada para prever categorias ou classes, como, por exemplo, se um cliente é provável de comprar um produto ou não. O objetivo da classificação é alocar instâncias em categorias predefinidas com base nos padrões observados nos dados de treino.

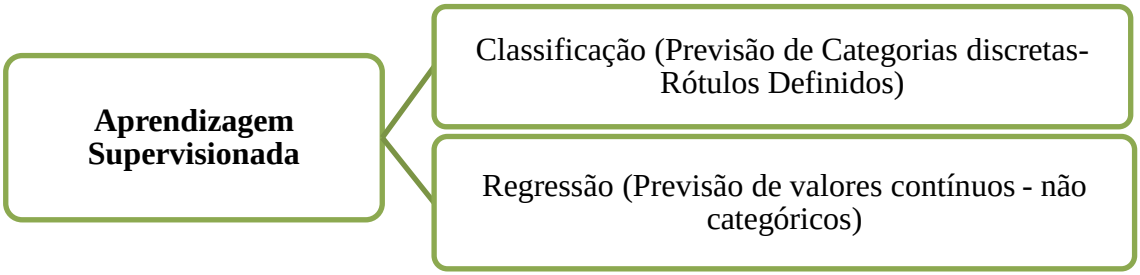


Figura 4. A aprendizagem supervisionada classificada em duas categorias

2.4.2. Tipos mais comuns de algoritmos de aprendizagem supervisionada

Os algoritmos de aprendizagem supervisionada podem ser classificados em dois grandes grupos: algoritmos de regressão, utilizados para prever valores contínuos, e algoritmos de classificação, utilizados para prever categorias ou classes (O'Reilly, 2025). Existem modelos que podem ser usados em ambos os contextos, com pequenas adaptações. Abaixo apresenta-se alguns dos algoritmos mais utilizados em cada uma dessas categorias:

Algoritmos de Regressão

- **Regressão Linear:** Algoritmo simples de aprendizagem supervisionada, utilizado para prever valores contínuos com base na relação linear entre variáveis de entrada e saída. É amplamente aplicado em problemas de regressão, ajustando um modelo que encontra a linha de melhor ajuste aos dados. Para isso, estima coeficientes, como a inclinação (β1) e a intercepção (β0), de forma a minimizar os erros quadráticos entre os valores reais e as previsões do modelo (Koray, Gyimah, Metwally, Rahnema, & Tomomewo, 2025).

- **Árvores de Decisão:** Algoritmo de regressão que prevê valores contínuos ao minimizar a soma dos erros quadráticos. Inicia-se com um nó raiz, que é particionado recursivamente até não haver melhoria significativa ou as instâncias

nos nós serem insuficientes. Cada nó calcula um valor médio predito, reduzindo os erros quadráticos. A qualidade preditiva é avaliada pelo coeficiente R^2, considerado fiável se acima de 10% (IBM, 2025).

- ***Random Forest*:** Técnica baseada na combinação de múltiplas árvores de decisão, agregando as suas previsões para gerar um modelo mais robusto e preciso. Pode ser utilizado tanto para classificação como para regressão, sendo que, no caso da regressão, o modelo calcula a média das previsões de cada árvore individual, minimizando o impacto de variações nos dados e aumentando a estabilidade dos resultados (Francisco, 2024).

- ***Support Vector Regression* (SVR):** Técnica derivada do *Support vector machines* (SVM) que prevê valores contínuos ao aproximar os valores reais dentro de um intervalo de tolerância (ϵ), minimizando o erro e mantendo a generalização. Usa funções *kernel* para captar relações não lineares e destaca-se pela resistência a outliers e melhor generalização face à regressão tradicional (Li, Tan, Zeng, Su, & Qiao, 2025).

Algoritmos de Classificação

- **Regressão Logística:** Algoritmo de classificação binária, extensível a multiclasse, que estima a probabilidade de uma instância pertencer a uma classe, usando a função sigmoide para limitar as previsões a [0,1]. Ajusta os pesos das variáveis com base nos dados de treino e utiliza um limiar de decisão para classificar as instâncias (Haque & Sarker, 2025).

- **Árvores de Decisão:** Algoritmo não paramétrico usado para classificação e regressão, que organiza os dados numa estrutura hierárquica, dividindo-os recursivamente em subconjuntos com base em regras extraídas do treino. Em classificação, aprende regras de decisão para prever a classe da variável-alvo (Scikit-learn Developers, 2025).

- ***Random Forest*:** Algoritmo de classificação por *ensemble learning*, que treina múltiplas árvores de decisão em subconjuntos aleatórios dos dados e combina as

previsões para maior robustez e precisão. Usa votação maioritária para classificar e reduz a variabilidade, tornando o modelo mais resistente a outliers e menos suscetível a *overfitting* (Francisco, 2024).

- ***Support Vector Classification* (SVC):** Modelo baseado em SVM para classificação binária e multiclasse, que procura um híperplano ótimo para maximizar a margem entre classes. Suporta vários *kernels* (linear, polinomial, RBF e sigmoide) para modelar relações não lineares. Utiliza validação cruzada aninhada para ajustar hiperparâmetros e melhorar a generalização (Scikit-learn Developers, 2025).

- **K-*Nearest Neighbors* (KNN):** Algoritmo de classificação e regressão que usa a proximidade entre pontos, assumindo que exemplos semelhantes têm características semelhantes. Na classificação, atribui a classe com base no voto dos k vizinhos mais próximos. Na regressão, prevê pela média dos valores dos vizinhos. Simples e intuitivo, mas lento em grandes conjuntos de dados devido ao cálculo de distâncias para todas as amostras (IBM, 2025).

2.4.3. Vantagens da Aprendizagem Supervisionada

A aprendizagem supervisionada destaca-se pela alta precisão na previsão de padrões e na tomada de decisões baseadas em dados, graças ao uso de dados rotulados no treino, que permite aprender relações entrada-saída de forma eficaz. É versátil, aplicando-se a classificação, regressão, reconhecimento de imagem e processamento de linguagem natural, com métricas bem estabelecidas (*Accuracy*, *Precision*, *Recall* e F1-*score*, para classificação) para avaliar o desempenho. Além disso, adapta-se a novos dados, melhorando continuamente a precisão das previsões, e oferece flexibilidade na definição de classes e adaptação às necessidades da tarefa, o que aumenta a sua aplicabilidade em diferentes contextos (Geeks for Geeks, 2024).

2.4.4. Desvantagens da Aprendizagem Supervisionada

Apesar das vantagens da aprendizagem supervisionada, esta apresenta limitações a considerar na formulação de problemas, recolha de dados e escolha do modelo (Supervised and Unsupervised learning, 2023).

Uma das principais limitações é o *overfitting*, que ocorre quando o modelo memoriza padrões específicos dos dados de treino, em vez de identificar relações generalizáveis, resultando em baixo desempenho em novos dados. Para mitigar este problema, utilizam-se técnicas de regularização (L1 Lasso e L2 *Ridge*) para limitar a complexidade do modelo e selecionar atributos relevantes. A validação cruzada é também aplicada para avaliar o desempenho em diferentes subconjuntos, reduzindo o risco de *overfitting* (Obaido, et al., 2024).

Outro desafio é a seleção de atributos (*Feature Selection*), essencial para identificar variáveis relevantes. Este processo pode ser moroso e requer conhecimento do domínio, pois escolhas inadequadas comprometem o desempenho do modelo. Técnicas como o *Recursive Feature Elimination* (RFE), que remove variáveis irrelevantes, e o *Random Forest Feature Importance*, que avalia a contribuição de cada variável, ajudam a reduzir a dimensionalidade, minimizar *overfitting* e melhorar a eficiência computacional, tornando os modelos mais interpretáveis e robustos (Ustebay, Turgut, & Aydin, 2018).

A aprendizagem supervisionada também enfrenta desafios com grandes volumes de dados, devido ao alto custo computacional e à necessidade de infraestrutura robusta. Além disso, depende de dados rotulados, cuja rotulagem manual é dispendiosa e morosa, especialmente em domínios que exigem especialistas. Quando há escassez de dados rotulados, a viabilidade desta abordagem é limitada. Para ultrapassar estas limitações, utilizam-se estratégias alternativas como a aprendizagem semi-supervisionada, que combina dados rotulados e não rotulados, e o *data augmentation*, que gera novos exemplos para aumentar a diversidade do treino e reduzir o *overfitting*. Estas técnicas melhoram a robustez e a generalização dos modelos (Machado, 2022).

2.5. Aprendizagem Não Supervisionada

A aprendizagem não supervisionada é um ramo da aprendizagem automática que utiliza dados não rotulados. Diferencia-se da aprendizagem supervisionada, pois não possui categorias ou resultados específicos. O seu principal objetivo é identificar padrões e relações intrínsecas nos dados, sem conhecimento prévio sobre o seu significado. É particularmente poderosa na análise exploratória de dados, permitindo compreender estruturas subjacentes e relações entre variáveis. Assim, desempenha um papel essencial na descoberta de conhecimento, deteção de agrupamentos naturais e extração de *insights* ocultos em grandes volumes de dados.

Os modelos de aprendizagem não supervisionada trabalham com dados não rotulados e utilizam algoritmos específicos para analisar, agrupar e identificar padrões, sem necessitar de instruções explícitas, conforme representado na Figura 5. Ao contrário da aprendizagem supervisionada, onde o modelo é treinado com dados anotados, aqui o modelo recebe apenas valores de entrada e deve descobrir autonomamente as estruturas e agrupamentos latentes nos dados (Unsupervised Learning, 2023).

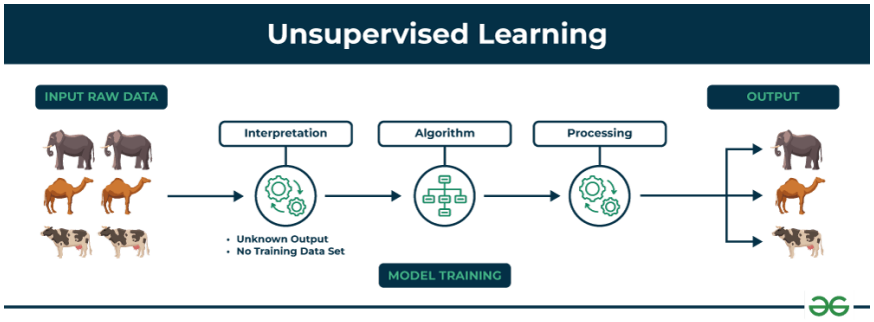


Figura 5. Aprendizagem não-supervisionada

Fonte: Unsupervised Learning, 2023

Os *inputs* utilizados nos modelos de aprendizagem não supervisionada variam consoante a natureza do problema e os objetivos da análise. De um modo geral, estes modelos operam com:

- **Dados não estruturados:** Podem conter informações ruidosas, como valores irrelevantes, dados em falta ou informações desconhecidas, o que dificulta a extração de *insights*. No entanto, quando bem interpretado, o ruído pode fornecer contexto adicional e enriquecer a análise.

- **Dados não rotulados:** Na ausência de valores-alvo predefinidos, a aprendizagem não supervisionada procura identificar padrões ou estruturas subjacentes nos dados. Este tipo de abordagem é particularmente útil em cenários onde a rotulagem manual é impraticável ou dispendiosa, como na análise de comportamento de utilizadores, segmentação de clientes ou agrupamento de imagens.

Deste modo, a escolha da aprendizagem não supervisionada não depende apenas da ausência de rótulos, mas da necessidade de descobrir padrões e relações ocultas, obtendo conhecimento útil de forma autónoma.

2.5.1. Tipos de algoritmos de aprendizagem não supervisionada

A aprendizagem supervisionada pode ser dividida em duas categorias principais: *Clustering* e *Association Rule Learning*. Cada uma destas categorias utiliza diferentes abordagens e algoritmos:

- ***Clustering*:** Identifica agrupamentos naturais num conjunto de dados multidimensional, usando medidas de similaridade como a distância Euclidiana, sem precisar de rótulos definidos. É usado em segmentação de clientes, deteção de anomalias, compressão de dados, quantização de imagens e mineração de dados. Cada *cluster* é representado por um centroide, que é o ponto médio das instâncias agrupadas. Contudo, a identificação de *clusters* pode ser desafiante, pois podem ter formas e tamanhos variados, complicando a segmentação em padrões não supervisionados (Omran, Engelbrecht, & Salman, 2007).

- ***Association Rule Learning*:** Identifica relações e dependências em grandes conjuntos de dados, onde os itens são armazenados em transações extraídas de bases de dados ou geradas por processos externos. Devido à sua boa escalabilidade, é uma ferramenta essencial na extração de conhecimento e na tomada de decisões. Aplica-se em análise de cestas de compras, *cross selling*, deteção de produtos estratégicos (*loss-leader*), entre outros (Cios, Swiniarski, Pedrycz, & Kurgan, 2007).

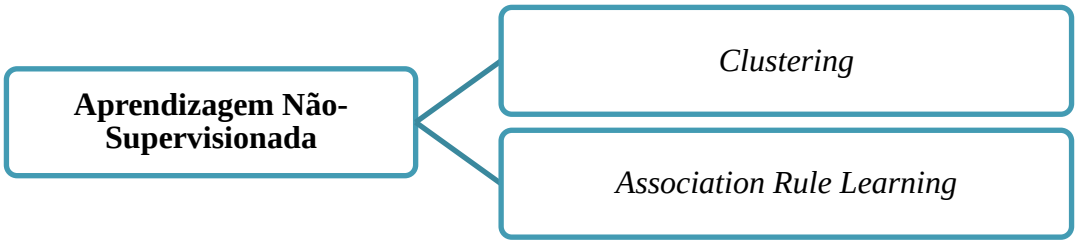


Figura 6. A aprendizagem não-supervisionada classificada em duas categorias

2.5.2. Tipos mais comuns de algoritmos de aprendizagem não-supervisionada

Algoritmos de *Clustering*

- **K-*means Clustering*:** Algoritmo de agrupamento que minimiza a distância Intra *cluster*, atribuindo cada ponto de dados ao *cluster* com o centroide mais próximo. O processo inicia-se com a seleção de K centroides, seguida da atribuição das instâncias ao *cluster* correspondente. Os centroides são então recalculados iterativamente, repetindo-se o processo até à convergência, ou seja, até que deixem de sofrer alterações significativas (Omran, Engelbrecht, & Salman, 2007).

- ***Hierarchical Clustering*:** Método que identifica comunidades em redes, sem impor uma ordenação linear. Constrói uma estrutura hierárquica de *clusters*, permitindo visualizar as relações entre eles num dendrograma. Pode ser aglomerativo (*clusters* fundem-se progressivamente) ou divisivo (*clusters* são divididos iterativamente). No *clustering* aglomerativo, destacam-se o *single-linkage* (menor distância entre nós) e o *centroid linkage* (distância entre centróides) (Lane, Maiocco, Bhatia, & Climer, 2020).

- ***Density-Based Clustering* (DBSCAN):** Identifica *clusters* com base na densidade dos pontos de dados, sendo eficaz para descobrir formas arbitrárias de *clusters* e lidar com ruído (Ester, Kriegel, Sander, & Xu, 1996).

- ***Mean-Shift Clustering*:** Algoritmo de agrupamento não paramétrico que desloca os pontos de dados em direção aos centroides, representando a média dos vizinhos. Também conhecido como *mode-seeking algorithm*, destaca-se por não

exigir a definição prévia do número de *clusters*, ajustando-os automaticamente com base na largura de banda (*bandwidth*) (Ranjbarzadeh, Caputo, Tirkolaee, Ghoushchi, & Bendechache, 2023).

- ***Spectral Clustering*:** Método de agrupamento baseado em grafos, que usa a decomposição espectral para identificar similaridades entre os nós. Destaca-se pelo seu desempenho superior em problemas onde métodos clássicos falham (Theodoridis & Koutroumbas, 2009).

Association Rule Learning

- **Algoritmo *Apriori*:** Um dos algoritmos mais conhecidos, utiliza pesquisa em largura para contar o suporte dos conjuntos de itens e gerar candidatos. Baseia-se na propriedade de fecho descendente (*downward closure*) e segue uma abordagem de baixo para cima (*bottom-up*), permitindo identificar itens frequentes e extrair regras de associação. No entanto, o seu desempenho reduz-se com bases de dados maiores, pois exige múltiplas varreduras das transações, tornando-se menos eficiente para grandes volumes de dados (Narvekar & Syed, 2015).

- **Algoritmo FP-*Growth*:** Método de mineração de itens frequentes que dispensa a geração de candidatos. Em vez da estratégia "gerar e testar" do *Apriori*, utiliza a FP-*tree*, uma estrutura compacta que comprime a base de dados e permite crescer fragmentos de padrões frequentes, tornando o processo mais eficiente. Segue uma abordagem dividir para conquistar: primeiro, comprime a base de dados numa FP-*tree*, preservando as associações entre itens. Depois, divide-a em subconjuntos condicionais, onde cada um corresponde a um item frequente, permitindo minerar padrões de forma independente. Esta abordagem é mais eficiente que o *Apriori*, pois evita múltiplas varreduras da base de dados e reduz a complexidade da mineração em grandes volumes de dados (Han, Kamber, & Pei, 2012).

- **Algoritmo Eclat (*Mining frequent itemsets using the vertical data format)*:** Converte um conjunto de transações do formato horizontal (*TID-itemset*) para o formato vertical (*item-TID set*) e realiza a mineração por interseção de conjuntos

TID, utilizando a propriedade Apriori e otimizações como *diffset* (Han, Kamber, & Pei, 2012). Segue uma abordagem *bottom-up*, analisando apenas o banco de dados vertical e efetuando uma única varredura, sem necessidade de múltiplas leituras. Calcula o suporte dos conjuntos de itens, mas não a confiança das regras de associação. É um algoritmo simples e eficiente para identificar itens frequentes, sendo especialmente útil em grandes bases de dados (Bansal, 2014).

2.5.3. Desafios da Aprendizagem Não Supervisionada

A avaliação do desempenho de algoritmos de aprendizagem não supervisionada representa um desafio significativo devido à ausência de rótulos ou categorias predefinidas nos dados. Compreender o processo de tomada de decisão dos modelos resultantes é complexo, dificultando a interpretação dos resultados. Além disso, há o risco de *overfitting*, onde os algoritmos podem ajustar-se excessivamente ao conjunto de dados de treino específico, comprometendo a sua capacidade de generalização para novos dados. A qualidade dos dados de entrada é crucial, pois os algoritmos de aprendizagem não supervisionada são sensíveis a dados ruidosos ou incompletos, o que pode levar a resultados imprecisos ou falsos. Alguns algoritmos também podem ser computacionalmente caros, especialmente quando lidam com conjuntos de dados de alta dimensão ou grandes volumes de dados. Esses desafios destacam a complexidade envolvida na aplicação eficaz da aprendizagem não supervisionada (Unsupervised Learning, 2023).

2.5.4. Vantagens da Aprendizagem Não-Supervisionada

A aprendizagem não supervisionada apresenta várias vantagens significativas. Em primeiro lugar, não requer que os dados de treino estejam rotulados, o que simplifica o processo de preparação de dados e elimina a necessidade de rotulagem manual. Além disso, a redução de dimensionalidade pode ser facilmente realizada utilizando técnicas de aprendizagem não supervisionada, permitindo simplificar conjuntos de dados complexos e melhorar a eficiência dos algoritmos de ML.

A capacidade de descobrir padrões previamente desconhecidos nos dados é outra vantagem importante, proporcionando *insights* valiosos e revelando informações ocultas. Isso também permite obter *insights* de dados não rotulados que, de outra forma, poderiam

ser difíceis ou impossíveis de obter, sendo especialmente útil em situações em que os dados estão disponíveis, mas a rotulagem manual é inviável. Adicionalmente, a aprendizagem não supervisionada é eficaz na identificação de padrões e relacionamentos nos dados sem instruções prévias sobre o que procurar, o que pode levar à descoberta de novos conhecimentos e uma compreensão mais profunda dos dados (Supervised and Unsupervised learning, 2023).

2.5.5. Desvantagens da Aprendizagem Não-Supervisionada

Apesar das suas vantagens, a aprendizagem não supervisionada apresenta algumas desvantagens significativas. Em primeiro lugar, pode ser difícil avaliar o desempenho dos algoritmos, uma vez que não existem rótulos ou categorias predefinidas para comparar os resultados. Além disso, a interpretação do processo de tomada de decisão dos modelos de aprendizagem não supervisionada pode ser complexa e desafiadora.

Outra desvantagem é a sensibilidade à qualidade dos dados de entrada; dados ruidosos ou incompletos podem levar a resultados falsos ou imprecisos. Além disso, alguns algoritmos de aprendizagem não supervisionada, especialmente aqueles que lidam com dados de alta dimensão ou grandes conjuntos de dados, podem ser computacionalmente caros, representando um desafio em termos de recursos computacionais e tempo de processamento.

A falta de respostas predefinidas durante o treino torna difícil medir a precisão ou eficácia dos modelos. Frequentemente, os resultados têm uma precisão inferior em comparação com os modelos de aprendizagem supervisionada. Ademais, o utilizador pode precisar investir tempo na interpretação e rotulagem das classes que resultam dessa classificação, o que pode ser uma tarefa laboriosa e requerer especialização no domínio dos dados (Supervised and Unsupervised learning, 2023).

2.6. Etapas de Pré-Processamento de ML

A etapa de processamento de ML refere-se a um conjunto de tarefas que envolvem a preparação e o tratamento de dados para a construção e treino de modelos de ML. Essas tarefas são essenciais para garantir que os dados estejam no formato adequado e de qualidade para que os modelos possam aprender com precisão a partir deles.

Pré-Processamento de dados

O pré-processamento de dados representa um dos desafios mais complexos em projetos de Aprendizagem Automática. Cada conjunto de dados é único e altamente específico para o projeto em questão. No entanto, há semelhanças suficientes entre projetos de modelação preditiva que permitem criar uma sequência aproximada de etapas e subtarefas que provavelmente serão executadas. Este processo proporciona um enquadramento no qual se pode considerar a preparação de dados necessária para o projeto, tendo em conta tanto a definição do projeto realizada antes da preparação de dados quanto a avaliação de algoritmos de aprendizagem automática realizada posteriormente (Brownlee, 2020).

O pré-processamento de dados é uma etapa essencial na aprendizagem automática, pois prepara os dados brutos para que possam ser utilizados de forma eficaz nas fases seguintes. Este processo pode incluir adição, eliminação ou transformação de dados, garantindo que a informação esteja limpa, coerente e estruturada antes de ser usada pelos algoritmos. Os dados, tal como são recolhidos, muitas vezes contêm ruído, valores em falta e inconsistências, o que pode comprometer a qualidade dos resultados. Para evitar isso, o pré-processamento melhora a eficiência e facilita a extração de padrões, tornando os modelos mais precisos e fiáveis. Sem um bom pré-processamento, até os algoritmos mais avançados podem falhar, pois os modelos aprendem com base na qualidade da informação que recebem. Por isso, esta fase é considerada uma das mais críticas no processo de ML (Brownlee, 2020).

Seleção de Atributos

Antes de iniciar o pré-processamento de dados detalhado, uma etapa inicial fundamental é a seleção de atributos (colunas). Esta fase permite ao utilizador determinar quais atributos (colunas) serão utilizadas no modelo, baseando-se na relevância, qualidade e no contexto do problema em questão. Um aspeto crucial neste processo é a avaliação criteriosa da qualidade das variáveis selecionadas, garantindo que contribuem para um melhor desempenho do modelo e facilitam a interpretação dos resultados (Guyon, Nikravesh, Gunn, & Zadeh, 2008). A seleção de atributos pode incluir:

- **Relevância dos Dados:** Deve-se identificar e utilizar apenas as variáveis significativas, evitando a inclusão de dados desnecessários que possam introduzir ruído e comprometer a eficácia do modelo.

- **Redução da Dimensionalidade:** Embora a seleção de atributos se centre na escolha das variáveis mais relevantes, podem ser aplicadas técnicas como a Análise de Componentes Principais (PCA, *Principal Component Analysis*), que transforma as variáveis originais em componentes principais. No entanto, ao contrário da seleção de atributos, o PCA pode reduzir a interpretabilidade dos dados, uma vez que gera novas variáveis sem um significado direto (Jaadi, 2024).
- **Facilidade de Interpretação:** A escolha adequada dos atributos simplifica a análise e a compreensão dos resultados, mantendo as variáveis com um significado claro e interpretável, algo que não acontece com técnicas como o PCA.

Esta etapa é particularmente relevante em grandes volumes de dados, onde muitas variáveis podem não impactar significativamente o desempenho do modelo. Ao focar nos atributos realmente importantes, não só se obtém uma melhor precisão, como também se facilita a interpretação dos resultados, contribuindo para tomadas de decisão mais informadas.

Após a seleção das variáveis relevantes, procede-se à categorização dos atributos, distinguindo entre variáveis categóricas e variáveis numéricas, assegurando que cada uma seja tratada da forma mais adequada no modelo.

Categorização de Atributos

Uma etapa crítica no pré-processamento de dados é a categorização de variáveis, que envolve identificar quais variáveis são categóricas e quais são numéricas. Essa distinção é vital, pois diferentes algoritmos de ML lidam de maneiras distintas com esses tipos de dados.

Atributos Categóricos:

Os atributos categóricos representam categorias ou grupos e podem ser divididos em duas subcategorias:

- **Nominais:** Não possuem uma ordem específica (ex: cor, tipo de fruta).
- **Ordinais:** Apresentam uma ordem lógica (ex: níveis de satisfação: baixo, médio, alto).

Atributos Numéricos:

Os atributos numéricos, por outro lado, podem ser medidos e representam quantidades. Estes podem ser categorizadas da seguinte forma:

- **Inteiros (*int*):** Números inteiros sem parte decimal (ex: 1, 2, 3).
- **Flutuantes (*float*):** Números com parte decimal (ex: 3.14, 2.71).
- **Complexos (*complex*):** Números que têm uma parte real e uma parte imaginária (ex: 2 + 3i).
- **Decimais (*decimal*):** Números que são usados em cálculos de precisão, como em transações financeiras.
- **Fracionários (*frac*):** Representações fracionárias de números (ex: 1/2, 3/4).

Os atributos **Booleanos (*bool*)** (*true/false*) são frequentemente considerados categóricos, mas, em termos computacionais, podem ser convertidos em valores numéricos (0 e 1) para facilitar o processamento, especialmente em algoritmos que utilizam codificação binária (Choudhury, 2024).

A correta categorização dos atributos é essencial para garantir a compatibilidade entre os dados e os algoritmos de aprendizagem automática (ML), assegurando que as técnicas de modelação sejam adequadas ao tipo de dados disponíveis. Algoritmos de classificação, como KNN e Random Forest, possuem estratégias específicas para lidar com atributos categóricos, enquanto algoritmos de regressão, como RLS e SVR, requerem atributos numéricos para realizar previsões com precisão. Além disso, a categorização dos dados facilita o pré-processamento, permitindo transformações como a conversão de variáveis categóricas em variáveis *dummy (one-hot encoding)* ou a normalização e padronização de variáveis numéricas, otimizando assim o desempenho do modelo durante o treino (Fontelles, 2024).

Discretização de Variáveis

Após a seleção das colunas e a categorização de variáveis, uma etapa importante que se pode realizar é a discretização de variáveis numéricas. A discretização é o processo de transformar variáveis contínuas em intervalos discretos, ou "*bins*". Este processo é especialmente útil para variáveis como idade, onde pode ser benéfico agrupar dados em categorias, como "10-20", "20-30", "30-40", etc. Mas irá sempre depender do tipo de

problema. Os utilizadores podem especificar os intervalos desejados para a discretização, fornecendo os limites (*bins*) para a variável "Idade". Além disso, têm a opção de definir rótulos (*labels*) para cada intervalo, facilitando a interpretação dos resultados (Alexandropoulos, Kotsiantis, & Vrahatis, 2019).

Exemplo de discretização para a variável "Idade":

- *Digite os bins para Idade (separados por vírgulas): 0, 18, 35, 65*
- *Digite os labels para Idade (separados por vírgulas): Jovem, Adulto, Idoso*

Essa abordagem não só simplifica a análise, mas também pode melhorar o desempenho de alguns modelos ao reduzir a complexidade dos dados.

Limpeza de Dados (*Data Cleaning*)

Os dados frequentemente apresentam falhas, ruídos e inconsistências, o que torna a análise difícil. A limpeza de dados é crucial para preencher lacunas, suavizar ruídos, identificar erros e corrigir inconsistências. Esses problemas são comuns em bases de dados reais, frequentemente decorrentes de erros humanos ou de computador ao inserir dados, falhas na transmissão de dados ou limitações tecnológicas. A limpeza de dados garante que os dados estejam prontos para análise, evitando confusões no processo de *data mining.* Embora muitos processos de ML tenham procedimentos para lidar com dados incompletos ou ruidosos, nem sempre são robustos. Portanto, é útil pré-processar os dados com procedimentos de limpeza (A. Sivakumar, 2017).

A limpeza de dados é uma etapa importante no processo de ML, pois envolve identificar e eliminar dados omissos, duplicados ou irrelevantes. O objetivo é garantir que os dados sejam precisos, consistentes e sem erros, uma vez que dados imprecisos podem afetar negativamente o desempenho do modelo. A limpeza de dados é fundamental para assegurar a precisão e confiabilidade dos conjuntos de dados. Sem uma limpeza adequada, imprecisões, erros e inconsistências podem comprometer a validade dos resultados analíticos. Além disso, dados limpos facilitam a modelação e o reconhecimento de padrões. Em resumo, a limpeza de dados é essencial para preparar conjuntos de dados confiáveis e de alta qualidade, que são a base para análises precisas e para a tomada de decisões informadas (Geeks for Geek, 2024).

- **Etapas importantes para efetuar a limpeza de dados**

Realizar a limpeza de dados envolve um processo sistemático para identificar e corrigir erros, inconsistências e imprecisões num conjunto de dados.

- **Remoção de Observações Indesejadas:** Identificar e eliminar observações irrelevantes ou redundantes do conjunto de dados. Essa etapa envolve analisar minuciosamente as entradas de dados para encontrar informações que não contribuem significativamente para a análise. A remoção de observações indesejadas otimiza o conjunto de dados, reduzindo o ruído e melhorando a qualidade geral.

- **Correção de Erros de Estrutura:** Abordar problemas estruturais no conjunto de dados, como inconsistências nos formatos de dados, convenções de nomenclatura ou tipos de variáveis. Padronizar formatos e corrigir discrepâncias melhora a consistência dos dados e facilita análises e interpretações precisas.

- **Tratamento de Dados Ausentes:** Desenvolver estratégias para lidar com dados ausentes pode envolver a imputação de valores com base em métodos estatísticos, a remoção de dados com valores ausentes ou o uso de técnicas avançadas de imputação. Estes valores podem ser substituídos pela moda (valor mais frequente), por um valor constante escolhido pelo utilizador, pela mediana (no caso de variáveis numéricas), podem ser mantidos ou excluídos. Tudo depende do objetivo do utilizador.

- **Tratamento de *Outliers*:** Identificar e gerir outliers, ou seja, instâncias que se desviam significativamente do comportamento usual, é um passo crucial. Consoante o contexto, deve-se avaliar se esses pontos devem ser removidos ou ajustados para minimizar o impacto na análise. Este processo é essencial para obter resultados mais precisos e confiáveis a partir dos dados. Os valores atípicos podem ser substituídos pela média, pela mediana, ajustados aos limites definidos (alterando os outliers para que se mantenham dentro de valores preestabelecidos), ou podem ser removidos, seja em conjunto ou apenas os mais extremos. Alternativamente, podem ser mantidos, dependendo do objetivo do utilizador. A

escolha da abordagem a adotar depende sempre do contexto e do propósito da análise

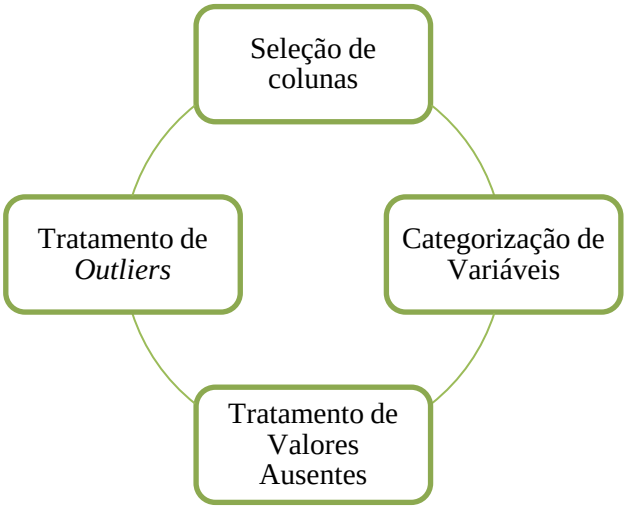


Figura 7. ***Data Cleaning***

Normalização e Padronização

Após a limpeza de dados, a normalização e a padronização são etapas cruciais no pré-processamento, especialmente para algoritmos sensíveis à escala dos dados, como KNN, SVM e Redes Neurais. Estas transformações garantem que as diferentes variáveis estejam na mesma escala, evitando que variáveis com magnitudes maiores dominem aquelas com magnitudes menores, o que poderia prejudicar o desempenho dos modelos.

- **Normalização:** A normalização, também conhecida como *Min-Max Scaling*, ajusta os valores dos dados para que fiquem dentro de uma escala específica, normalmente entre 0 e 1. Este método facilita a comparação entre variáveis com diferentes magnitudes e é particularmente útil em algoritmos baseados em distâncias, como o KNN e o K-*Means*. A normalização é recomendada quando os dados apresentam outliers significativos, uma vez que comprime a amplitude dos valores e torna os dados mais homogéneos para análise (Bhandari, 2024).

- **Padronização:** A padronização transforma os dados de forma que tenham média 0 e desvio padrão 1. Esta técnica é preferida quando os dados seguem uma distribuição normal, pois mantém a forma da distribuição e altera apenas a escala. A padronização é amplamente utilizada em algoritmos como Regressão Linear e

SVC, que podem beneficiar deste procedimento para convergência mais rápida e desempenho superior (Scikit-learn developers, 2025).

De acordo com a documentação do *Scikit-learn* (Scikit-learn developers, 2025) tanto a normalização quanto a padronização são essenciais para garantir que os algoritmos funcionem eficientemente em espaços de alta dimensionalidade, prevenindo que uma variável com maior amplitude domine as outras devido à sua escala. Estas técnicas de escalonamento ajudam a evitar *overfitting*, especialmente em modelos que dependem fortemente da comparação de distâncias, como o KNN e o SVM. Na prática, a escolha entre normalização e padronização depende da natureza dos dados e do algoritmo utilizado.

Codificação de Variáveis Categóricas

Após a normalização e padronização, é fundamental realizar a codificação das variáveis categóricas para um formato que os modelos de ML possam entender. Variáveis categóricas, como "cor", "cidade" ou "categoria de produto", precisam ser transformadas em uma representação numérica, pois a maioria dos algoritmos de ML requer que todos os dados sejam numéricos. As técnicas mais comuns de codificação incluem:

- ***One-Hot Encoding***: Este método cria variáveis binárias para cada categoria de uma variável categórica. Cada categoria é transformada em uma nova coluna, e o valor 1 ou 0 indica a presença ou ausência dessa categoria na observação. Este método é amplamente utilizado quando as categorias não possuem uma ordem intrínseca (nominais). No entanto, pode aumentar significativamente o número de colunas no conjunto de dados, especialmente quando há muitas categorias possíveis (Scikit-learn developers, 2025).
- ***Label Encoding:*** Atribui um número inteiro a cada categoria, permitindo que o modelo aprenda uma ordem numérica. Este método é mais adequado quando as categorias possuem uma ordem natural, como "baixo", "médio" e "alto". Embora seja eficiente, o seu uso em variáveis sem uma hierarquia clara pode induzir o modelo a interpretar relações entre as categorias que não existem (Scikit-learn developers, 2025).

A escolha entre esses métodos depende da natureza da variável categórica e do algoritmo que está sendo utilizado. Enquanto o *One-Hot Encoding* é preferido para variáveis sem

uma ordem natural, o *Label Encoding* pode ser vantajoso em situações onde as categorias têm uma hierarquia implícita ou explícita. Além disso, a escalabilidade pode ser um fator determinante, pois o *One-Hot Encoding* pode levar à explosão dimensional, especialmente em grandes conjuntos de dados categóricos.

A codificação correta das variáveis categóricas pode impactar significativamente o desempenho do modelo, e por isso é essencial entender a estrutura dos dados antes de optar por uma técnica específica (Kaggle, 2024).

2.7.*Grid Search*

A otimização de hiperparâmetros é um processo crítico para maximizar o desempenho dos modelos de ML. Hiperparâmetros, como o número de vizinhos no KNN ou o número de árvores no *Random Forest*, controlam o comportamento do algoritmo e não são ajustados automaticamente durante o treino.

A Pesquisa em Grelha (*Grid Search*) é uma técnica sistemática de otimização de hiperparâmetros que explora todas as combinações possíveis dentro de um espaço pré-definido, avaliando cada configuração com base em métricas específicas, como por exemplo *Accuracy*, *Precision*, *Recall, F1-Score* e *Mean Squared Error*(MSE) (Bergstra & Bengio, 2012).

Os hiperparâmetros são valores que controlam o comportamento dos algoritmos de aprendizagem automática, mas que não podem ser ajustados diretamente durante o treino do modelo. Exemplos incluem o número de vizinhos no algoritmo KNN ou o número de árvores no *Random Forest.*

A técnica *Grid Search* utiliza validação cruzada para garantir que as combinações testadas sejam avaliadas de forma justa e reduzam o risco de *overfitting*. Para isso, divide-se o conjunto de treino em subconjuntos menores, aplicando os modelos treinados em diferentes partes dos dados.

2.7.1. Etapas do *Grid Search*

- **Definir o Espaço de Hiperparâmetros:** Selecionar os hiperparâmetros a serem ajustados e especificar os valores possíveis para cada um.

- **Construir Combinações:** Criar uma matriz com todas as combinações possíveis de hiperparâmetros.
- **Treinar e Avaliar Modelos:** Testar cada combinação utilizando validação cruzada para avaliar o desempenho.
- **Selecionar a Melhor Combinação:** Escolher os hiperparâmetros que geraram o melhor modelo, de acordo com a métrica de desempenho definida para a avaliação, ou seja, os valores dos hiperparâmetros são selecionados com base no desempenho do modelo medido por essa métrica.

Embora o *Grid Search* garanta a exploração completa do espaço de procura, pode ser computacionalmente dispendioso para problemas com múltiplos parâmetros e grandes volumes de dados. Métodos alternativos, como *Random Search* e *Otimização Bayesiana*, são frequentemente utilizados para melhorar a eficiência em situações mais complexas (Bergstra & Bengio, 2012).

2.7.2. Vantagens do *Grid Search*

- Garantia de encontrar a melhor combinação de hiperparâmetros, se esta existir dentro do espaço de procura selecionado, proporcionando uma exploração completa do espaço de hiperparâmetros.
- Simplicidade e transparência do processo, facilitando a interpretação dos resultados e o entendimento do comportamento do modelo.

2.7.3. Desvantagens do *Grid Search*

- Elevado custo computacional, especialmente em espaços de procura grandes, o que torna o método ineficiente para problemas com muitos hiperparâmetros ou conjuntos de dados extensos.
- Ineficiente quando os hiperparâmetros não são igualmente importantes, já que o *Grid Search* trata todas as combinações com a mesma prioridade, sem considerar a relevância relativa de cada hiperparâmetro.

2.7.4. Alternativas

- ***Random Search*:** Técnica de ajuste de hiperparâmetros que funciona através da seleção aleatória de valores dentro de um espaço de procura previamente definido, avaliando depois o desempenho do modelo para cada combinação escolhida. Trata-se de um método simples e eficiente, permitindo explorar um grande número de possibilidades com menos iterações do que o *Grid Search*, que testa exaustivamente todas as combinações possíveis. Esta abordagem é especialmente vantajosa quando o espaço de procura é vasto e a relação entre os hiperparâmetros e o desempenho do modelo é difícil de prever (Youness, Phan, & Boulakia, 2023). Nestes casos, o *Random Search* possibilita uma exploração mais rápida e flexível, frequentemente identificando boas combinações sem necessidade de testar todas as opções, o que reduz o custo computacional e torna o processo mais eficiente (Bergstra & Bengio, 2012).

- **Otimização Bayesiana:** A otimização bayesiana (BO) é uma técnica baseada no Teorema de *Bayes*, utilizada para otimizar funções objetivo com custos de avaliação elevados. Reduz o número de tentativas necessárias para ajustar hiperparâmetros, tornando-se mais eficiente do que métodos como *Grid Search* e *Random Search* (Youness, Phan, & Boulakia, 2023). O seu funcionamento assenta na criação de um modelo probabilístico da função objetivo, que é atualizado com novas avaliações, permitindo ao algoritmo selecionar os melhores hiperparâmetros com base nos resultados anteriores (Snoek, Larochelle, & Adams, 2012). Uma abordagem avançada, o *Tree-structured Parzen Estimator* (TPE), recorre a uma estrutura em árvore para dividir o espaço de hiperparâmetros em regiões de alta e baixa probabilidade, avaliando a função objetivo nos pontos mais promissores dentro das regiões de maior probabilidade.

Tabela 2. Vantagens e Desvantagens dos métodos de otimização

Método	Vantagens	Desvantagens	Computacionalmente
Grid Search	Garantido para encontrar a melhor combinação se existir dentro do espaço de procura selecionado. Simples e transparente.	Caro, especialmente com espaços de procura grandes. Ineficiente quando os hiperparâmetros não são igualmente importantes. Limitado a valores de hiperparâmetros predefinidos.	Caro, especialmente com espaços de procura grandes.
Random Search	Eficiente em termos de tempo de computação. Melhor exploração do espaço de procura em comparação com o *Grid Search*.	Não garante encontrar a melhor combinação. A aleatoriedade pode levar a soluções sub-ótimas em alguns casos.	Eficiente em termos de tempo de computação.
Bayesian Search	Exploração eficiente e inteligente do espaço de procura. Geralmente requer menos iterações para encontrar bons hiperparâmetros.	Mais complexo de implementar do que o *Grid* e o *Random Search*. Pode exigir o ajuste de parâmetros adicionais para o modelo probabilístico.	Geralmente requer menos iterações para encontrar bons hiperparâmetros.

Fonte: Adaptado de Youness, Phan, & Boulakia, 2023

2.8.Divisão de Dados em Treino e Teste

A divisão de dados é uma etapa essencial no desenvolvimento e avaliação de modelos de ML, pois garante uma avaliação justa do modelo, reduzindo o risco de *overfitting* e melhorando a capacidade de generalização para novos dados. Uma das abordagens mais comuns é a divisão percentual *(percentage split*), onde os dados são separados em conjuntos de treino e teste de acordo com uma proporção predefinida (Bishop, 2006). Tipicamente, utiliza-se um rácio de 80:20, permitindo avaliar a capacidade do modelo de generalizar para novos dados (Geeks for Geeks, 2024). No entanto, este método de percentagem *split* é apenas uma das estratégias utilizadas para avaliar modelos.

Uma alternativa amplamente utilizada é a validação cruzada (*cross-validation*), que melhora a fiabilidade da avaliação do modelo (Hastie, Tibshirani, & Friedman, 2009). Um dos métodos mais comuns é a validação cruzada k-*fold*, onde os dados são divididos em k subconjuntos (*folds*). O modelo é treinado k vezes, utilizando k-1 *folds* para treino e o *fold* restante para teste, garantindo que cada subconjunto seja utilizado como teste

pelo menos uma vez. Esta abordagem reduz a variabilidade dos resultados e melhora a capacidade de generalização do modelo (Salimi, Ghobrial, & Bonakdari, 2024).

Outra técnica relevante é a LOOCV (*leave-one-out cross-validation)*, uma variação extrema do k-*fold*, onde cada instância dos dados é usada como conjunto de teste, enquanto todas as outras instâncias são usadas para treino. Embora forneça uma avaliação robusta, pode ser computacionalmente exigente para grandes volumes de dados (Zhang, Theeramunkong, & Khamsemanan, 2024). Exige a criação de N modelos para N instâncias.

A escolha entre *split*, *cross-validation* ou LOOCV depende do tamanho do conjunto de dados, dos recursos computacionais disponíveis e do objetivo do modelo.

2.9.Modelos em ML

Um modelo de *ML* é um sistema computacional projetado para identificar padrões e tomar decisões com base em dados previamente analisados. Esses modelos são treinados em conjuntos de dados históricos, permitindo-lhes aprender as relações entre variáveis e, posteriormente, generalizar esse conhecimento para dados desconhecidos. O processo de treino envolve a otimização de parâmetros e estruturas, resultando num modelo capaz de fazer previsões ou classificações precisas (Databricks, 2024).

Os modelos de ML são aplicáveis a uma ampla gama de cenários, incluindo processamento de linguagem natural (NLP), reconhecimento de imagens, análise de séries temporais e deteção de fraudes. Por exemplo, no NLP, um modelo pode ser treinado para reconhecer intenções em frases e identificar sentimentos. Em tarefas de reconhecimento de imagem, os modelos podem ser utilizados para identificar objetos, como veículos e animais, com elevada precisão. Durante o processo de treino, os algoritmos de ML analisam dados rotulados (Aprendizagem Supervisionada) ou não rotulados (Aprendizagem Não-Supervisionada) para extrair padrões. Este processo resulta num modelo otimizado, armazenado em estruturas computacionais específicas, que pode ser aplicado para processar novos dados e gerar previsões relevantes. A escolha do modelo adequado depende do tipo de problema a ser resolvido. Para problemas de classificação, utilizam-se modelos que agrupam os dados em categorias distintas. Para previsões de valores contínuos, recorrem-se aos modelos de regressão. Em contrapartida,

quando o objetivo é identificar grupos dentro de dados não rotulados, empregam-se técnicas de *Clustering* (QuinnRadich, V-alje, & Eliotcowley, 2024).

Entre os critérios para determinar a adequação do uso de ML estão a necessidade de decisões ou avaliações consistentes e automatizadas, a dificuldade em descrever explicitamente as regras ou critérios para tomadas de decisão e a disponibilidade de dados rotulados que podem ser utilizados para treinar os modelos. Estes modelos, uma vez treinados, são capazes de generalizar padrões e comportamentos, permitindo a sua aplicação em cenários complexos onde as regras manuais seriam difíceis de implementar. Além disso, ferramentas avançadas de ML permitem a personalização de algoritmos e otimização de parâmetros, garantindo eficiência e adaptabilidade para diferentes contextos.

Os modelos de ML podem ser categorizados em diferentes categorias, dependendo do tipo de problema que resolvem. Nesta secção, apresenta-se quatro tipos de categorias, sendo elas a Classificação, Regressão, *Clustering* e Associação.

2.9.1. Modelos de Classificação

Os modelos de classificação são amplamente utilizados em problemas de previsão de categorias discretas ou rótulos. O objetivo principal é atribuir uma observação a uma categoria específica com base nas características fornecidas. Esta abordagem é aplicada em diversos domínios, incluindo deteção de fraudes, diagnóstico médico, análise de sentimentos e reconhecimento de imagens.

Em termos técnicos, os modelos de classificação baseiam-se em dados de treino previamente rotulados, permitindo que o algoritmo aprenda as relações entre variáveis de entrada (*features*) e os rótulos de saída. Após o treino, o modelo é avaliado com dados novos ou não vistos, a fim de medir a sua capacidade de generalização.

A precisão e eficiência dos modelos de classificação dependem da seleção do algoritmo, da preparação adequada dos dados e da otimização de hiperparâmetros. Diversos algoritmos são amplamente utilizados, como K-*Nearest Neighbors* (KNN), *Random Forest* e *Support Vector Machines* (SVM).

K-Nearest Neighbors (KNN)

O K-*Nearest Neighbors* (KNN) é um algoritmo simples e intuitivo utilizado em ML para problemas de classificação e regressão. Baseia-se na ideia de que objetos semelhantes se encontram próximos uns dos outros no espaço multidimensional. Esse algoritmo é amplamente utilizado devido à sua simplicidade, eficácia em pequenas bases de dados e flexibilidade para diferentes tipos de problemas. Este modelo classifica uma observação com base na categoria mais comum entre os seus "k" vizinhos mais próximos (Sun, Du, & Shi, 2018).

Funcionamento do Algoritmo:

O algoritmo KNN segue um processo de três etapas principais para classificar uma nova observação:

1. **Cálculo da Distância:**

    - Mede a distância entre a nova observação e cada ponto no conjunto de treino.

    - As métricas de distância mais comuns são (Geeks for geeks, 2024):

        - **Distância Euclidiana** – Adequada para dados contínuos.

$$\boldsymbol{distance}(\boldsymbol{x}, X_i) = \sqrt{\sum_{j=1}^{d} = (x_j - X_{ij})^2}$$

Onde:

- $\boldsymbol{x}$: Vetor que representa as coordenadas ou características do ponto de dados de teste.
- X_i : Vetor que representa as coordenadas ou características de um ponto no conjunto de treino. Este ponto é comparado com o ponto $\boldsymbol{x}$**.**
- x_j: Valor da j-ésima característica do ponto $\boldsymbol{x}$ (ponto de teste).

- X_{ij}: Valor da j-ésima característica do ponto X_i (ponto do conjunto de treino) que está sendo comparado com o ponto $\boldsymbol{x}$.

- **Distância Manhattan** – Mais robusta para diferentes escalas e valores esparsos.

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

Onde:

- x, y: Vetores que representam os dois pontos cujas distâncias estão a ser calculadas.
- x_i, y_i: As i-ésimas características de x e y, ou seja, os valores de cada dimensão.
- $x_i - y_i$: A diferença absoluta entre os valores das i-ésimas características de x e y.

- **Distância de Minkowski** – Generaliza as anteriores.

$$\boldsymbol{distance}(\boldsymbol{x}, y) = (\sum_{i=1}^{n} = (x_i - y_i)^p)^{\frac{\mathbf{1}}{\boldsymbol{p}}}$$

Onde:

- x, y: Vetores que representam os dois pontos cujas distâncias estão a ser calculadas, geralmente um ponto de treino e um ponto de teste.
- x_i, y_i: As i-ésimas características dos pontos x e y, ou seja, os valores de cada dimensão de x e y.
- $(x_i - y_i)^p$: A diferença elevada à potência p entre as características correspondentes dos pontos x e y. Para *p=1*, obtém-se a distância de Manhattan, e para *p=2*, obtém-se a distância Euclidiana.
- p: O parâmetro que define a "ordem" da distância.

2. **Seleção dos "k" Vizinhos Mais Próximos:**

    - Seleciona os *k* pontos com as menores distâncias em relação ao ponto de teste.

    - O valor de *k* é um hiperparâmetro e deve ser ajustado para equilibrar *overfitting* e *underfitting*.

3. **Classificação por Votação Maioritária:**

    - A nova observação é classificada de acordo com a classe mais frequente entre os vizinhos (*k* mais próximos).

    - Para regressão, o valor atribuído é a média ou mediana dos vizinhos selecionados.

Vantagens:

- Simplicidade e Implementação Intuitiva: Não requer ajuste inicial complexo.
- Adaptável a Pequenos Conjuntos de Dados: Funciona bem para dados de pequena dimensão.

Desvantagens:

- Sensibilidade a *Outliers*: Valores extremos podem distorcer os resultados.
- Ineficiente em Grandes Volumes de Dados: Torna-se computacionalmente dispendioso devido à necessidade de calcular distâncias para todas as observações.
- Dependência de Escalas: Pode exigir normalização dos dados para garantir resultados precisos.

Aplicações Comuns (IBM, 2025)**:**

- **Diagnóstico médico:** Utilizado na classificação de pacientes para prever doenças e avaliar riscos, como no diagnóstico de ataques cardíacos e cancro da próstata.
- **Deteção de anomalias:** Aplicado na identificação de fraudes e comportamentos atípicos em transações bancárias e negociações financeiras.

- **Reconhecimento facial e análise de imagens:** Usado para reconhecimento facial e na classificação de objetos e padrões visuais em imagens.
- **Sistemas de recomendação:** Empregado na personalização de recomendações, sugerindo produtos, músicas ou filmes com base nas preferências dos utilizadores.
- **Pré-processamento de dados:** Auxilia na imputação de valores ausentes, preenchendo lacunas com base na média ou nos vizinhos mais próximos.
- **Setor financeiro:** Aplicado na avaliação de risco de crédito e previsão de mercado, ajudando a determinar a fiabilidade de crédito de indivíduos ou empresas.

Hiperparâmetros Comuns:

Tabela 3. Hiperparâmetros Comuns KNN

Hiperparâmetro	Intervalo / Opções Comuns	Padrão
Número de vizinhos (***n_neighbors***)	Número Inteiro (1-20)	5
Ponderação dos vizinhos (***weights***)	'*uniform'*, *'distance'*, ou função personalizada	*'uniform'*
Algoritmo de procura (***algorithm***)	auto', 'ball_tree', 'kd_tree', 'brute'	'auto'
Tamanho da folha (***leaf_size***)	Número Inteiro	30
Parâmetro da métrica de *Minkowski* (p)	1 (Manhattan), 2 (Euclidiana) ou outro valor	2
Métrica de distância (***metric***)	*minkowski'*, *'euclidean'*, *'manhattan'*, entre outras	*'minkowski'*
Parâmetros adicionais da métrica (***metric_params***)	Dicionário de parâmetros específicos da métrica	*None*
Número de jobs paralelos (***n_jobs***)	None (1 thread) ou -1 (todas as *threads* disponíveis)	*None*

Fonte: Scikit-learn Developers, 2025

Random Forest

O *Random Forest* é um algoritmo baseado no conceito de *ensemble learning* onde múltiplos modelos (neste caso, árvores de decisão) trabalham em conjunto para produzir previsões mais precisas e robustas. Foi introduzido por Breiman (2001) e é amplamente utilizado para tarefas de classificação e regressão devido à sua simplicidade, versatilidade e elevada precisão. Estudos recentes, com o de Hamza & Sjarif (2024), demonstram a aplicação prática do *Random Forest* em áreas como análise de sinais biomédicos, consolidando a sua relevância na era moderna de *Machine Learning.*

O *Random Forest* é uma abordagem poderosa para problemas de classificação e regressão. A sua capacidade de lidar com grandes volumes de dados, combinada com resistência a outliers e flexibilidade em lidar com variáveis complexas, torna-o amplamente adotado em diversos domínios.

Apesar das suas vantagens, deve-se considerar a necessidade de otimizar hiperparâmetros e utilizar técnicas de validação cruzada para evitar *overfitting*. Ferramentas como *Grid Search* podem ser usadas para ajustar os parâmetros do modelo.

Funcionamento do Algoritmo

O *Random Forest* constrói um conjunto de árvores de decisão independentes e combina os seus resultados para obter uma previsão mais estável e precisa:

1. ***Bootstrap Sampling (Bagging*)**: Cria subconjuntos aleatórios do conjunto de treino com reposição (amostragem *bootstrap*).

2. **Construção de Árvores de Decisão:** Em cada subconjunto, constrói-se uma árvore de decisão, dividindo os nós com base na métrica de impureza (por exemplo, Gini Index[9] ou Entropia[10]).

[9] Medida de impureza usada em árvores de decisão, avaliando a probabilidade de um elemento ser classificado incorretamente. Quanto maior o seu valor, maior a heterogeneidade dos dados e o risco de erros na classificação (Rizvi, 2024).
[10] Medida de impureza que quantifica o grau de desordem ou incerteza num conjunto de dados. Em árvores de decisão, tal como o Índice de Gini, é utilizada para determinar a melhor forma de dividir um nó, de modo a obter subconjuntos mais homogéneos (Rizvi, 2024).

3. **Combinação dos Resultados:** Para classificação, usa a votação maioritária entre as árvores.

4. **Agregação Final:** As previsões individuais das árvores são combinadas para formar o resultado.

Vantagens:

- **Alta Precisão e Robustez:** Reduz o risco de *overfitting* devido à combinação de múltiplas árvores.
- **Versatilidade:** Suporta variáveis categóricas e contínuas, além de lidar bem com dados não lineares.
- **Resistência a Outliers e Dados Omissos:** É menos sensível a valores extremos devido ao efeito de agregação.
- **Funcionalidade Integrada para Seleção de Variáveis:** Mede a importância das variáveis (*feature importance*), facilitando a interpretação.
- **Escalabilidade:** Funciona bem com grandes volumes de dados e pode ser paralelizado para otimizar o desempenho.

Desvantagens:

- **Complexidade Computacional Elevada:** Requer mais memória e poder computacional em comparação com modelos simples como árvores de decisão únicas.
- **Menor Interpretabilidade:** Embora as árvores individuais sejam interpretáveis, o modelo global é mais difícil de explicar.
- **Sensibilidade a Dados Desequilibrados:** Pode ser necessário ajustar os pesos das classes para lidar com dados não balanceados.

Aplicações Comuns:

- Diagnóstico médico (Identificação de padrões em exames médicos e deteção precoce de doenças).
- Deteção de fraudes financeiras.
- Modelação Climática e Previsão Meteorológica.
- Análise de Sentimentos.

Parâmetros Importantes no *Random Forest*

- ***n_estimators*:** Número de árvores no modelo (valores mais altos reduzem o *overfitting*).
- ***max_features*:** Número de *features* consideradas para divisão em cada nó (por padrão, raiz quadrada das *features*).
- ***max_depth*:** Profundidade máxima de cada árvore (evita *overfitting*).
- ***min_samples_split*:** Número mínimo de amostras necessárias para dividir um nó.
- ***min_samples_leaf***: Número mínimo de amostras necessárias em cada folha.
- ***Bootstrap***: Ativa/desativa a amostragem *bootstrap*.

Tabela 4. Hiperparâmetros comuns *Random Forest*

Hiperparâmetro	Intervalo / Opções Comuns	Padrão
Número de árvores (*n_estimators*)	Número inteiro (50-500)	100
Profundidade máxima (*max_depth*)	Número inteiro ou None	None
Amostras mínimas para dividir (*min_samples_split*)	Número inteiro (2-6)	2
Amostras mínimas por folha (*min_samples_leaf*)	Número inteiro	1
Características máximas por divisão (*max_features*)	‘auto’, 'sqrt', 'log2' ou *None*	'sqrt'
Bootstrap (*bootstrap*)	*True* ou *False*	True

Fonte: (Hyperparameters of Random Forest Classifier, 2021)

Support Vector Classifier (SVC)

O *Support Vector Classifier* (SVC) é uma implementação específica do algoritmo *Support Vector Machines* (SVM), amplamente reconhecido pela sua eficácia na resolução de problemas de classificação. Este modelo distingue-se pela capacidade de lidar com dados de alta dimensionalidade e pela flexibilidade em modelar relações não lineares, o que o torna particularmente útil em situações onde outros algoritmos apresentam limitações (Brandt, Benedek, Guerin, & Fliege, 2020).

O SVC é baseado no conceito de encontrar um híper plano ótimo que separe as diferentes classes de dados, maximizando a margem entre elas. A margem representa a distância entre o híper plano e os pontos de dados mais próximos, denominados vetores de suporte.

Estes vetores são fundamentais para definir o limite de decisão e garantir que o modelo seja robusto e generalizável para novos dados.

Se os dados não forem linearmente separáveis no espaço original, o SVC utiliza funções de *kernel* para projetar os dados para um espaço de maior dimensão, onde a separação linear é possível. Esse processo é conhecido como transformação do espaço de características (*feature space transformation*) (Daza, Rueda, Sánchez, Espíritu, & Quiñones, 2024).

Funcionamento do Algoritmo

O SVC procura encontrar um híperplano ótimo que separe os dados em diferentes classes, maximizando a margem entre os pontos de dados mais próximos (chamados vetores de suporte) e o híper plano, conforme representado na Figura 8. Essa margem maximizada aumenta a capacidade do modelo de generalizar para novos dados.

1. **Transformação dos Dados:** Utiliza funções de *kernel* para transformar os dados para um espaço dimensional superior, onde possam ser separáveis linearmente.

2. **Definição da Margem Ótima:** Identifica os vetores de suporte mais próximos do híper plano, garantindo a separação ideal entre classes.

3. **Maximização da Margem:** Resolve um problema de otimização para maximizar a margem entre as classes.

4. **Classificação dos Dados:** Com base na posição relativa dos dados em relação ao híper plano.

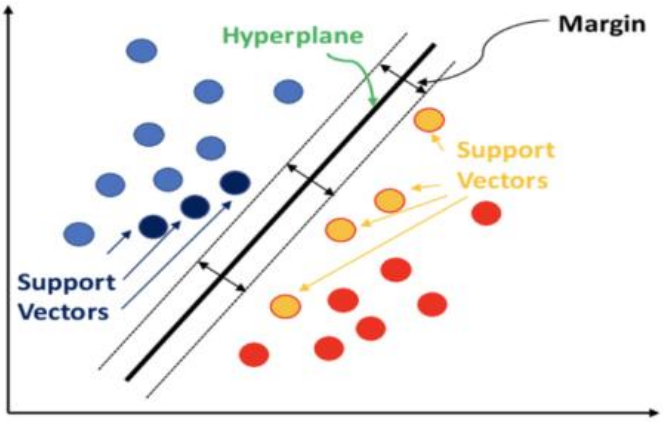


Figura 8. Funcionamento SVC
Fonte: Salunke, 2023

Vantagens:

- **Eficiência em Alta Dimensionalidade:** Funciona bem mesmo com dados complexos e espaços de alta dimensionalidade.
- **Versatilidade:** Suporta classificações lineares e não lineares, ajustando-se a diferentes problemas.
- **Resistência ao *Overfitting*:** Usa regularização para minimizar o *overfitting*, especialmente em pequenos conjuntos de dados.

Desvantagens:

- **Exigência Computacional Elevada:** Pode ser lento em grandes volumes de dados devido à necessidade de calcular matrizes complexas.
- **Dependência de Parâmetros:** Requer ajuste de hiperparâmetros, como o parâmetro C (controlo da margem) e *gamma* (escala do *kernel*).
- **Interpretação Difícil:** Modelos baseados em *kernels* não são tão interpretáveis quanto outros algoritmos, como árvores de decisão.

Aplicações Comuns:

- Análise de Sentimentos.
- Deteção de Fraudes.
- Reconhecimento de Linguagem Natural (NPL).
- Aplicações Médicas

Hiperparâmetros Comuns:

Tabela 5. Hiperparâmetros comuns SVC

Hiperparâmetro	Intervalo / Opções Comuns	Padrão
Função Kernel (*kernel*)	'linear', *'poly'*, 'rbf', *'sigmoid'*, *'precomputed'*	'rbf'
Parâmetro de Regularização (C)	Número real (>0), tipicamente 0.1, 1, 10	1.0
Coeficiente do Kernel (*gamma*)	*'scale'*, 'auto', ou valor numérico (>0)	*'scale'*
Tolerância ao Critério de Paragem (tol)	Número real (>0), geralmente 1e-3 ou 1e-4	1,00E-03
Parâmetro de Viés (degree)	Número inteiro ≥ 0 (ex: 2, 3, 4)	3
Coeficiente no *Kernel* Polinomial (coef0)	Número real (apenas para *kernel 'poly'* ou *'sigmoid'*)	0.0
Shrinkage (*shrinking*)	*True* ou *False*	*TRUE*
Probabilidade (*probability*)	*True* ou *False*	*FALSE*
Cache do Kernel (*cache_size*)	Número real (ex: 200)	200
Peso das Classes (*class_weight*)	*'balanced'* ou dicionário com pesos específicos	*None*
Iterações Máximas (*max_iter*)	Número inteiro ou -1 (sem limite)	-1
Forma da Função de Decisão (*decision_function_shape*)	'ovo', 'ovr'	'ovr'
Quebra de Empates (*break_ties*)	*True* ou *False*	*FALSE*
Estado Aleatório (*random_state*)	Número inteiro (ex: 42) ou *None*	*None*

Fonte: (Scikit-learn Developers, 2025)

Tipos de *Kernels* Utilizados no SVC

O SVC suporta diferentes funções de *kernel*, permitindo modelar relações complexas:

- **Linear:** Adequado para dados que podem ser separados por uma linha reta ou plano.
- **Polinomial:** Cria separações curvas ao elevar as entradas a potências específicas.
- ***Radial Basis Function* (RBF):** Popular para dados não lineares, cria divisórias curvas complexas.
- **Sigmoide:** Utilizado em redes neuronais e modelos com características similares.

Tipos de *Gamma* no SVC

O *gamma* é um hiperparâmetro essencial no SVC, especialmente ao utilizar o *kernel* RBF. Ele controla a influência de cada ponto de treino na criação do híperplano de decisão.

- ***'scale'* (Padrão):** Calcula automaticamente com base nos dados.
- ***'auto'*:** Define o *gamma* como 1 / número de características.
- **Personalizado**: Valor escolhido manualmente.

2.9.2. Modelos de Regressão

Os modelos de regressão são algoritmos amplamente utilizados para prever valores contínuos, modelando a relação entre variáveis independentes (*features*) e uma variável dependente (*target*) (Wang, Wang, Li, & Qi, 2024). Diferentemente dos modelos de classificação, que preveem categorias discretas, os modelos de regressão estimam valores numéricos. Estes modelos são aplicados em diversas áreas, como previsão de preços, análise financeira, crescimento populacional e análise de séries temporais (Duda, Woiciechowski, Scapini, & Soccol, 2024).

Nesta secção, são apresentados dois modelos utilizados na plataforma: RLS e SVR.

Tabela 6. Comparação entre RLS e SVR

Característica	RLS	SVR
Tipo de Relação	Linear	Linear e Não Linear
Interpretação	Simples e direta	Mais complexa, requer ajuste de parâmetros
Computacionalmente Eficiente	Elevada eficiência	Exige mais recursos computacionais
Tolerância a *Outliers*	Sensível a *outliers*	Controlada por margens
Escalabilidade	Boa para grandes conjuntos de dados	Mais adequado para pequenos volumes

Regressão Linear Simples (RLS)

A Regressão Linear Simples (RLS) é um dos métodos mais básicos e interpretáveis em *Machine Learning*. Assume uma relação linear entre as variáveis independentes e a variável dependente, ajustando uma linha reta que minimiza os erros entre os valores previstos e reais (Duda, Woiciechowski, Scapini, & Soccol, 2024).

Fórmula Matemática:

$$y = \beta_0 + \beta_1 x + \varepsilon$$

Onde:

- y: Variável dependente (valor previsto).
- x: Variável independente.
- β_0: Intercepto (valor inicial).
- β_1: Inclinação da reta (coeficiente).
- ε : Termo de erro residual (Wang, Wang, Li, & Qi, 2024)

Vantagens (Balgude, Gite1, Pradhan, & Lee, 2024):

- **Simplicidade:** Fácil de interpretar e implementar.
- **Rapidez:** Computacionalmente eficiente, mesmo com grandes volumes de dados.
- **Aplicabilidade:** Adequada para relações lineares simples

Desvantagens (Balgude, Gite1, Pradhan, & Lee, 2024):

- **Sensibilidade a *Outliers*:** Valores extremos podem influenciar fortemente o modelo.
- **Incapacidade de Modelar Relações Não Lineares:** Torna-se inadequada para dados complexos.

Aplicações Comuns (Zermane, Zermane, & Tohir, 2024):

- Previsão de vendas e preços.
- Estimativa de despesas médicas e imobiliárias.
- Avaliação de impacto de fatores específicos em processos industriais.

Hiperparâmetros Comuns:

O modelo de Regressão Linear Simples (RLS), na sua essência teórica, não possui hiperparâmetros ajustáveis. No entanto, ao usar a biblioteca *scikit-learn* para implementar este modelo, existem configurações adicionais que podem ser ajustadas, mas estas não são hiperparâmetros do modelo em si. Essas opções configuram como o modelo é ajustado ou pré-processado, como ajustar o intercepto ou normalizar as variáveis de entrada.

Tabela 7. Configurações comuns para RLS

Configuração	Descrição	Padrão
fit_intercept	Calcula o intercepto (β0). Se *False*, os dados devem estar centrados.	*TRUE*
copy_X	Copia x para evitar alterações nos dados originais.	*TRUE*
n_jobs	Número de *threads* para cálculos paralelos. -1 utiliza todos os processadores disponíveis.	*None*
positive	Força os coeficientes (β1) a serem positivos.	*FALSE*

Fonte: Scikit-learn Developers, 2025

Support Vector Regression (SVR)

O *Support Vector Regression* (SVR) é uma extensão do *Support Vector Machine* (SVM), adaptado para prever valores contínuos em vez de categorias. Este modelo diferencia-se ao utilizar *kernels* para modelar relações não lineares, oferecendo maior flexibilidade para dados complexos (Wang, Wang, Li, & Qi, 2024).

Princípio de Funcionamento: O SVR identifica um híper plano que minimiza os desvios em relação aos pontos de dados, permitindo certa margem de erro (ϵ) dentro da qual os valores previstos são considerados aceitáveis (Li, et al., 2024).

Funções de *Kernel* (Wang, Wang, Li, & Qi, 2024):

- **Linear**: Adequado para dados separáveis linearmente.
- **Polinomial**: Captura padrões mais complexos.
- ***Radial Basis Function*** (RBF): Modela relações altamente não lineares.

Vantagens (Zermane, Zermane, & Tohir, 2024):

- **Lida com Relações Não Lineares:** Graças à utilização de funções de *kernel*.
- **Eficaz com Pequenos Conjuntos de Dados:** Pode funcionar bem mesmo com menos exemplos de treino.
- **Robustez Contra Outliers Moderados:** Permite margem controlada para desvios.

Desvantagens (Wang, Wang, Li, & Qi, 2024):

- **Custo Computacional Elevado:** Requer mais tempo para treino e otimização, especialmente com grandes volumes de dados.
- **Dependência de Hiperparâmetros:** A precisão depende do ajuste adequado de parâmetros como C, *epsilon* e *gamma*.

Aplicações Comuns:

- **Previsão de Preços de Imóveis e Ações:** Modela relações complexas entre variáveis financeiras (Li, et al., 2024).
- **Análise de Séries Temporais:** Previsão de vendas e consumo energético (Zermane, Zermane, & Tohir, 2024).
- **Engenharia e Medicina:** Análise de dados experimentais e predição de propriedades físicas e químicas (Balgude, Gite1, Pradhan, & Lee, 2024).

Hiperparâmetros Comuns:

Tabela 8. Hiperparâmetros comuns SVR

Hiperparâmetro	Intervalo / Opções Comuns	Padrão
Função Kernel (*kernel*)	'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'	'rbf'
Parâmetro de Regularização (C)	Número real (>0), tipicamente 0.1, 1, 10	1.0
Coeficiente do *Kernel* (*gamma*)	*'scale'*, 'auto', ou valor numérico (>0)	*'scale'*
Tolerância ao Critério de Paragem (*tol*)	Número real (>0), geralmente 1e-3 ou 1e-4	1,00E-03
Parâmetro de Viés (*degree*)	Número inteiro ≥ 0 (apenas para *kernel 'poly'*)	3
Coeficiente no *Kernel* Polinomial (*coef0*)	Número real (apenas para *kernel 'poly'* ou *'sigmoid'*)	0.0
Epsilon (*epsilon*)	Número real (>0), como 0.1, 0.01	0.1
Shrinkage (*shrinking*)	*True* ou *False*	*TRUE*
Cache do Kernel (*cache_size*)	Número real (ex: 200)	200
Número de Iterações Máximas (*max_iter*)	Número inteiro ou -1 (sem limite)	-1

Fonte: Scikit-learn Developers, 2025

2.9.3. Modelos de Associação

Os modelos de associação são amplamente utilizados para identificar padrões ou regras frequentes em grandes volumes de dados. Este tipo de modelo é particularmente valioso em problemas onde é necessário descobrir relações ocultas entre variáveis, sendo amplamente aplicado em setores como o retalho, marketing e análise de redes sociais (Mostafa, Taha, & Eissa, 2024).

Os modelos de associação utilizam técnicas como Regras de Associação e algoritmos como o *Apriori* e o FP-*Growth* para derivar padrões. Estes métodos são fundamentais para tarefas como recomendações personalizadas, agrupamento de comportamentos de consumidores e deteção de anomalias (Katsumbe, Telukdarie, Munsamy, & Tshukudu, 2024).

Princípios Básicos das Regras de Associação

As Regras de Associação baseiam-se na identificação de conjuntos de itens frequentes em grandes bases de dados, com a finalidade de derivar regras que descrevam interações ou associações entre itens. Estas regras são expressas como:

$$A \rightarrow B$$

Onde A e B representam conjuntos de itens, e a regra indica que, sempre que A ocorre, há uma alta probabilidade de B também ocorrer.

Métricas Comuns:

- **Suporte**: Mede a frequência com que A e B aparecem juntos.
- **Confiança**: Mede a probabilidade condicional de B dado A.
- ***Lift***: Avalia a correlação entre A e B, indicando se a ocorrência conjunta é maior do que seria esperado ao acaso.
  (Tiwari & Park, 2024)

Aplicações Comuns dos Modelos de Associação

- **Recomendações Personalizadas:** Por exemplo, Sistemas de recomendação em plataformas de e-commerce, como "quem comprou X também comprou Y" (Mostafa, Taha, & Eissa, 2024).
- **Análise de Cestos de Compras:** Identificação de padrões de compra em supermercados para otimizar estratégias de marketing (Tiwari & Park, 2024).
- **Deteção de Fraudes:** Uso em transações financeiras para identificar comportamentos anômalos (Zuhdi, Sulistyanto, & Harahap, 2024).
- **Análise de Redes Sociais:** Descoberta de interações entre utilizadores em plataformas como Twitter ou Facebook.

Apriori

O *Apriori* é um dos algoritmos mais utilizados para descobrir regras de associação. Ele funciona de forma iterativa, gerando conjuntos de itens frequentes com base em um limite mínimo de suporte.

Funcionamento (Zuhdi, Sulistyanto, & Harahap, 2024):

- Criação de conjuntos de itens candidatos.
- Eliminação de conjuntos de itens abaixo do suporte mínimo.
- Construção de regras a partir dos conjuntos restantes.

Vantagens:

- Simples e eficiente para bases de dados de médio porte.
- Fácil de ajustar para diferentes cenários.

Desvantagens:

- Ineficiente com grandes bases de dados devido à necessidade de varrer os dados várias vezes.

Hiperparâmetros comuns:

Tabela 9. Hiperparâmetros comuns *Apriori*

Hiperparâmetro	Intervalo / Opções Comuns	Padrão
df	*DataFrame* codificado com valores 0/1 ou *True*/*False*	-
min_support	Número real [0.0, 1.0]	0.5
use_colnames	*True* ou *False*	*FALSE*
max_len	Número inteiro ≥ 1	*None*
verbose	Inteiro (>= 0)	0
low_memory	*True* ou *False*	*FALSE*

Fonte: Raschka, MLxtend, 2023

FP-*Growth (Frequent Pattern Growth)*

O FP-*Growth* é uma alternativa ao *Apriori*, que utiliza uma estrutura compacta de dados chamada FP-*Tree* para armazenar padrões frequentes.

Funcionamento (Zermane, Zermane, & Tohir, 2024):

- Construção da FP-*Tree* a partir da base de dados.
- Geração de conjuntos de itens frequentes diretamente da árvore.

Vantagens:

- Reduz a necessidade de múltiplas varreduras da base de dados.
- Mais eficiente em grandes volumes de dados.

Desvantagens:

- Exige maior memória para armazenar a FP-*Tree*.

Hiperparâmetros comuns:

Tabela 10. Hiperparâmetros comuns FP-*Growth*

Hiperparâmetro	Intervalo / Opções Comuns	Padrão
df	*DataFrame* codificado com valores 0/1 ou *True/False*	-
min_support	Número real [0.0, 1.0]	0.5
use_colnames	*True* ou *False*	*FALSE*
max_len	Número inteiro ≥ 1	*None*
verbose	Inteiro (>= 0), exibe progresso do algoritmo	0

Fonte: Raschka, 2023

2.9.4. Modelos de *Clustering*

Os modelos de *clustering* são algoritmos não supervisionados que agrupam dados em subconjuntos com base nas suas características. Cada grupo, ou *cluster*, é formado por elementos que apresentam maior semelhança entre si do que com elementos de outros grupos (Zermane, Zermane, & Tohir, 2024). Esses modelos são amplamente utilizados para segmentação de clientes, análise de dados exploratória e deteção de anomalias.

Dois algoritmos amplamente utilizados em *clustering* são o K-*Means* e o *Clustering* Hierárquico, que oferecem abordagens distintas para agrupar dados.

Tabela 11. Comparação entre K-*Means* e *Clustering* Hierárquico

Característica	K-Means	Clustering Hierárquico
Definição de *Clusters*	Necessária (k pré-definido)	Não requer número de *clusters*
Complexidade Computacional	Baixa	Alta
Visualização	Não oferece hierarquia	Representação detalhada com dendrograma
Sensibilidade a *Outliers*	Alta	Moderada

K-Means

O K-*Means* é um dos algoritmos mais populares devido à sua simplicidade e eficiência. Ele particiona o conjunto de dados em k *clusters*, com cada *cluster* representado pelo seu centroide.

Funcionamento:

- Escolha inicial de k centroides.
- Atribuição de cada ponto de dados ao *cluster* mais próximo com base numa métrica de distância (geralmente, distância Euclidiana).
- Atualização dos centroides como a média dos pontos atribuídos a cada *cluster*.
- Repetição até a convergência, ou seja, quando as atribuições não mudam significativamente (Katsumbe, Telukdarie, Munsamy, & Tshukudu, 2024).

Vantagens:

- Simplicidade de implementação.
- Escalabilidade para grandes conjuntos de dados.
- Alta eficiência em contextos onde os *clusters* são esféricos e bem separados.

Desvantagens:

- Requer a definição do número de *clusters* (k).
- **Sensível a valores iniciais dos centroides**: O K-*Means* tradicional pode ser altamente sensível à escolha inicial dos centroides, o que pode levar a uma convergência em soluções subótimas. Essa limitação é atenuada pelo K-*Means*++, que seleciona os centroides iniciais de maneira mais estratégica, garantindo uma distribuição mais equilibrada e, assim, aumentando a qualidade e a convergência do algoritmo (Mostafa, Taha, & Eissa, 2024).
- Sensível a *Outliers*.

Aplicações Comuns:

- Segmentação de clientes em marketing.
- Agrupamento de regiões em análises geográficas.
- Identificação de padrões em séries temporais (Wang, Wang, Li, & Qi, 2024).

Hiperparâmetros comuns:

Tabela 12. Hiperparâmetros comuns K-*means*

Hiperparâmetro	Intervalo / Opções Comuns	Padrão
n_clusters	Número inteiro (≥ 1), define o número de *clusters*	8
init	'k-*means*++', *'random'*, *callable* ou *array-like*	'k-*means*++'
n_init	'auto' ou Número inteiro (≥ 1), número de inicializações	'auto'
max_iter	Número inteiro (≥ 1), máximo de iterações por inicialização	300
tol	Número real (> 0), tolerância para critério de convergência	1,00E-04
random_state	Inteiro ou *None*, controla a semente aleatória	*None*
copy_x	*True* ou *False*, copia os dados de entrada	*TRUE*
algorithm	'lloyd', 'elkan'	'lloyd'

Fonte: Scikit-learn Developers, 2025

Clustering Hierárquico

O *Clustering* Hierárquico constrói uma hierarquia de *clusters*, que pode ser representada como uma árvore (dendrograma). Existem duas abordagens principais:

- **Aglomerativa:** Cada ponto inicia como um *cluster* individual, e os *clusters* são fundidos iterativamente com base em semelhanças.
- **Divisiva:** Começa com todos os pontos em um único *cluster* e divide-os sucessivamente (Balgude, Gite1, Pradhan, & Lee, 2024).

Funcionamento:

- Calcula as semelhanças entre pares de pontos usando métricas como distância Euclidiana ou de Manhattan.
- Combina os pontos mais semelhantes para formar *clusters* maiores ou dividir *clusters* maiores em menores.
- Continua até atingir o nível desejado de granularidade.

Vantagens:

- Não requer a definição prévia do número de *clusters*.
- Fornece uma visão detalhada da estrutura hierárquica dos dados.

Desvantagens:

- Menos eficiente em grandes conjuntos de dados.
- Resultados podem ser influenciados pela escolha da métrica de similaridade e método de ligação (por exemplo, média, simples, completa) (Ruan, Sun, Shou, & Wang, 2024).

Aplicações Comuns:

- Análise de redes sociais.
- Biologia computacional para agrupamento de genes ou proteínas.
- Classificação de documentos em bibliotecas digitais ((Ruan, Sun, Shou, & Wang, 2024).

Hiperparâmetros Comuns:

Tabela 13. Hiperparâmetros Comuns Clustering Hierárquico

Hiperparâmetro	Intervalo / Opções Comuns	Padrão
n_clusters	Número inteiro (≥ 1), define o número de clusters	2
metric	*'euclidean'*, 'l1', 'l2', *'manhattan'*, *'cosine'*, *'precomputed'*	*'euclidean'*
memory	*String* ou *None*, caminho para cache intermediário	*None*
connectivity	Matriz ou grafo de conectividade (*array*-like ou *sparse matrix*)	*None*
compute_full_tree	'auto', *True* ou *False*, decide se calcula a árvore completa	'auto'
linkage	*'ward'*, *'complete'*, *'average'*, *'single'*	*'ward'*
distance_threshold	Número real (> 0), distância máxima para formar *clusters*	None
compute_distances	*True* ou *False*, calcula distâncias para dendrogramas	*FALSE*

Fonte: Scikit-learn Developers, 2025

2.10. Métricas de Avaliação de Modelos em ML

As métricas de avaliação desempenham um papel crucial na análise de desempenho de modelos de *Machine Learning* (ML). Estas permitem aos cientistas de dados compreender a eficácia dos modelos e compará-los de forma consistente. As métricas variam consoante o tipo de problema a ser resolvido: classificação ou regressão.

2.10.1. Métricas para Problemas de Classificação

- ***Accuracy* (Taxa de Acerto):** Mede a proporção de previsões corretas em relação ao total de exemplos avaliados. É útil quando o conjunto de dados está balanceado e em situações em que erros de qualquer tipo (falsos positivos ou falsos negativos) têm um impacto semelhante. Um exemplo clássico é na classificação de imagens para identificar se uma foto contém ou não um objeto específico, como um carro (Filho, 2023).

  Fórmula:

$$Accuracy = \frac{VP + VN}{VP + VN + FP + FN}$$

  Onde:

  - **VP**: Verdadeiros Positivos.
  - **VN**: Verdadeiros Negativos.
  - **FP**: Falsos Positivos.
  - **FN**: Falsos Negativos.

- ***Precision* (Precisão):** Mede a proporção de verdadeiros positivos entre todas as previsões positivas feitas pelo modelo. É especialmente relevante em problemas onde os falsos positivos têm alto custo, como deteção de fraudes (Junior, 2023). Esta métrica é particularmente importante quando o custo de falsos positivos é elevado. Em outras palavras, ter um falso positivo representa um problema significativo. Um exemplo clássico disso é a classificação de emails entre Spam e Normais. Neste caso, é preferível que um email Spam acabe por ir para a Caixa de Entrada (Falso Negativo)

do que que um email importante seja enviado para a caixa de Spam (Falso Positivo) (Duarte, 2023).

Fórmula:

$$Precision = \frac{VP}{VP + FP}$$

Onde:

- **VP**: Verdadeiros Positivos.
- **FP**: Falsos Positivos.

- ***Recall* (Sensibilidade ou Revocação):** Avalia a proporção de casos positivos que o modelo conseguiu identificar corretamente. Este indicador é particularmente importante quando o custo de falsos negativos é elevado. Neste contexto, o *Recall* foca-se no oposto da *Precision*. Um exemplo típico de aplicação seria a deteção de fraudes em cartões de crédito. Neste tipo de situação, é preferível que o modelo bloqueie uma transação legítima (Falso Positivo) do que permitir uma transação fraudulenta (Falso Negativo) (Duarte, 2023).

Fórmula:

$$Recall = \frac{VP}{VP + FN}$$

Onde:

- **VP**: Verdadeiros Positivos.
- **FN**: Falsos Negativos.

- **F1-*Score*:** O F1-*Score* é a média harmónica entre a *Precision* e o *Recall*, oferecendo um equilíbrio entre essas duas métricas. Esta medida é especialmente útil quando se pretende considerar tanto os falsos positivos como os falsos negativos, ajudando a avaliar o desempenho do modelo de forma mais equilibrada (Duarte, 2023).

Fórmula:

$$\text{F1-Score} = 2 * \frac{Precision * Recall}{Precision + Recall}$$

2.10.2. Métricas para Problemas de Regressão

- **MAE - *Mean Absolute Error* (Erro Absoluto Médio):** Mede o erro médio absoluto entre os valores previstos e os reais, expressando a magnitude média das diferenças. O MAE é fácil de interpretar, pois mostra a média dos erros do modelo nas mesmas unidades da variável de destino, facilitando a compreensão do quão longe as previsões estão dos valores reais (Duarte, 2023).

  Fórmula:

$$MAE = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

  Onde:

  - y_i: Previsão.
  - x_i: Valor Real.
  - n: Número Total de Entradas.

- **MSE - *Mean Squared Error* (Erro Quadrático Médio):** É uma métrica comum que calcula a média dos quadrados das diferenças entre as previsões do modelo e os valores reais. Essa métrica atribui maior peso a erros maiores, tornando-se sensível a *outliers*. Quanto maior o MSE, pior é o desempenho do modelo, pois ele reflete o quão distantes estão as previsões dos valores reais (Duarte, 2023).

Fórmula:

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

Onde:

- n: Número Total de Entradas.
- Y_i: Valores Observados.
- $\hat{Y}_i$: Previsões

- **Coeficiente de Determinação (R^2):** É uma métrica que quantifica a proporção da variância na variável de destino explicada pelo modelo. O valor do R^2 varia de 0 a 1. Um valor próximo de 1 indica que o modelo explica uma grande parte da variabilidade nos dados, enquanto um valor próximo de 0 sugere que o modelo não está explicando muito. Em outras palavras, quanto mais próximo de 1 for o valor de R^2, melhor será a performance do modelo (Duarte, 2023).

  Fórmula (Azaank, 2020)**:**

$$R^2 = 1 - \frac{RSS}{TSS} <=> R^2 = 1 - \frac{\sum(y_{previsto} - \bar{y})^2}{\sum(y_{real} - \bar{y})^2}$$

  Onde:

  - RSS: Soma dos Quadrados dos Resíduos.
  - TSS: Soma dos Quadrados Totais.

2.10.3. Métricas para Problemas de Associação

As métricas para associação em *Machine Learning* são fundamentais para avaliar padrões entre variáveis ou eventos em conjuntos de dados. Estas métricas são amplamente utilizadas em tarefas de mineração de regras de associação e descoberta de relações significativas. Abaixo, são descritas as principais métricas:

- ***Support* (Suporte):** Mede a frequência com que um item ou conjunto de itens ocorre num *dataset*. Esta métrica ajuda a identificar associações comuns. Por exemplo, no setor de vendas, o suporte pode indicar quantas vezes produtos como "Bolachas" e "Sumo" são comprados juntos. Essa métrica é essencial para descartar associações esporádicas (Mlakar, Jr., & Fister, 2024).

  Fórmula:

$$Support\ (X \rightarrow Y) = \frac{Número\ de\ transações\ contendo\ X\ e\ Y}{Número\ de\ transações\ totais}$$

- ***Confidence* (Confiança):** É uma métrica que indica a proporção de vezes que, numa transação que contém o item A, também se encontra o item B. Ou seja, mede a

probabilidade de que o item B apareça numa transação, dado que o item A já está presente. Não é necessariamente igual ao suporte, já que esta se refere a uma relação condicional (se A ocorre, qual a probabilidade de B ocorrer) e não à frequência geral de A ou B nas transações (Costa, 2019).

Fórmula:

$$Confidence\ (X \rightarrow Y) = \frac{Número\ de\ transações\ contendo\ X\ e\ Y}{Número\ de\ transações\ contendo\ X}$$

- ***Lift* (Confiança):** Avalia a força de uma regra, ou seja, o grau em que a presença de um item A aumenta a probabilidade de ocorrência do item B em comparação com a probabilidade de B ocorrer independentemente de A (Anselmo, 2017).

  Fórmula:

$$Lift\ (X \rightarrow Y) = \frac{Confidence\ (X \rightarrow Y)}{Support(Y)}$$

  Onde:

  - $Confidence\ (X \rightarrow Y)$: Probabilidade de que, dado que A ocorre, B também ocorra.
  - $Support(Y)$: Proporção de transações que contêm o item B.

  Interpretação do valor:

  - ***Lift* = 1**: Indica que A e B são independentes, ou seja, a ocorrência de A não afeta a probabilidade de B ocorrer.
  - ***Lift* > 1**: Significa que A e B estão positivamente associados, ou seja, a ocorrência de A aumenta a probabilidade de B ocorrer.
  - ***Lift* < 1**: Indica que A e B estão negativamente associados, ou seja, a ocorrência de A diminui a probabilidade de B ocorrer.

2.10.4. Métricas para Problemas de *Clustering*

As métricas para *clustering* avaliam a qualidade de agrupamentos formados em tarefas de aprendizagem não supervisionada. Estas métricas podem ser divididas em duas categorias principais: métricas internas, que avaliam os agrupamentos com base nos dados utilizados, e métricas externas, que comparam os agrupamentos gerados com uma estrutura de referência.

- ***Silhouette Score*:** É uma métrica utilizada para avaliar a qualidade dos resultados de *clustering* em agrupamento de dados. Este índice é calculado medindo a semelhança de cada ponto de dados com o *cluster* a que pertence e a sua diferença em relação aos outros *clusters*. Esta é comumente utilizada para avaliar o desempenho de algoritmos de *clustering*, como o K-*Means* (Gültekin, 2023).

  Fórmula:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

  Onde:

  - $a(i)$: Média da distância entre o ponto i e todos os outros pontos dentro do mesmo *cluster*. Em outras palavras, é a coesão do ponto i dentro do seu próprio *cluster*. Este valor representa o grau de semelhança do ponto de dados em relação aos outros no seu *cluster*.
  - $b(i)$: Média da distância entre o ponto i e todos os pontos de qualquer outro *cluster*. Ou seja, é a separação do ponto i em relação ao *cluster* mais próximo. Este valor indica a diferença entre o ponto de dados e os pontos de dados nos outros *clusters*.

  Interpretação do valor:

- **$s(i)$ próximo de 1:** Indica que o ponto está bem agrupado, ou seja, está próximo do centro do seu próprio *cluster* e distante dos outros *clusters*, apontando para bons resultados de agrupamento.

- $\boldsymbol{s(i)}$ **próximo de 0:** Sugere *clusters* sobrepostos ou pontos de dados igualmente próximos de vários *clusters*.

- $\boldsymbol{s(i)}$ **negativo:** Indica que o ponto foi provavelmente atribuído ao *cluster* errado, pois está mais próximo de outro cluster, apontando para maus resultados de agrupamento.

- ***Davies-Bouldin Index:*** É uma métrica usada para avaliar a qualidade de um agrupamento (*clustering*). Ele mede a "distância" entre os *clusters* e a sua coesão interna. O índice é utilizado para comparar a dispersão Intra *cluster* com a separação entre *clusters*. Em termos simples, quanto menor o índice, melhor a qualidade do *clustering*, pois significa que os *clusters* são mais compactos e bem separados (GeeksforGeeks, 2023).

  Fórmula:

$$DB = \frac{1}{k}\sum_{i=1}^{k} \max\left(\frac{S_i + S_j}{d(c_i, c_j)}\right)$$

  Onde:

  - k: Número total de *clusters*.
  - S: É a medida da coesão (ou dispersão) do *cluster* i, geralmente calculada como a média das distâncias entre os pontos dentro do *cluster*. Ou seja, é a distância média entre cada instância do grupo e o seu centroide.
  - d: É a distância entre os centroides dos grupos, ou seja, a separação entre os *clusters*.

  Interpretação do valor:

- **Índice baixo**: Indica que os *clusters* são bem separados e compactos, ou seja, têm boa qualidade.

- **Índice alto**: Indica que os *clusters* estão mal definidos, sendo menos compactos ou mal separados.

- ***Calinski-Harabasz Index*:** É também conhecido como índice de variância entre e dentro dos *clusters*. É uma métrica utilizada para avaliar a qualidade de um agrupamento, e é calculado a partir da relação entre a dispersão entre os *clusters* e a dispersão dentro dos *clusters*. O índice mede o quão bem separados estão os *clusters*, e quanto mais distante os *clusters* estão uns dos outros em comparação à variabilidade interna, melhor é o desempenho do modelo de *clustering*. O Índice de *Calinski-Harabasz* tende a aumentar com o número de *clusters*, então é mais útil para comparar diferentes números de *clusters* e escolher aquele que resulta no melhor agrupamento. Quanto maior o índice, melhor será a qualidade do *clustering* (Wei, 2020).

Fórmula:

$$CH = \frac{Tr(B_{k)}}{Tr(W_{k)}} * \frac{n-k}{k-1}$$

Onde:

- n: Número total de pontos de dados.
- k: Número de *Cluster*s.
- $Tr(B_{k)}$: Traço da matriz de dispersão entre os *clusters*, que mede a dispersão das médias dos *clusters* em relação à média global.
- $Tr(W_{k)}$: Traço da matriz de dispersão dentro dos *clusters*, que mede a dispersão dos pontos dentro de cada *cluster*.

Interpretação do valor:

- **Índice baixo**: Indica que os *clusters* são mal definidos, com pouca separação entre eles e alta dispersão dentro dos *clusters*.
- **Índice alto**: Quanto maior o valor do índice, melhor a separação entre os *clusters* e a compactação dos mesmos, ou seja, os *clusters* são bem definidos.

2.11. Streamlit

A escolha da *framework* Streamlit para o desenvolvimento da plataforma MLCASE foi motivada pela sua simplicidade, flexibilidade e capacidade de criar interfaces interativas para aplicações de *Machine Learning*. O Streamlit é uma *framework* de código aberto amplamente utilizada para transformar *scripts* de Python em aplicações web interativas, com foco em usabilidade e rapidez de desenvolvimento.

2.11.1. Principais Vantagens do Streamlit

- **Desenvolvimento Rápido e Intuitivo:** O Streamlit permite construir aplicações completas com poucas linhas de código, tornando o desenvolvimento de protótipos e aplicações funcionais extremamente rápido. Essa característica é ideal para um projeto como o MLCASE, que procura combinar eficiência com facilidade de uso. Esta particularidade foi destacada em aplicações como a previsão de elegibilidade de crédito (Kiwa, Mnkandla, Ndlovu, Dube, & Nyoni, 2024) e análise de dados médicos (Silva, et al., 2024).

- **Visualizações Dinâmicas:** Oferece suporte nativo para gráficos interativos, tabelas dinâmicas e visualizações de dados em tempo real, como demonstrado no estudo de Sundararaj (2024), que utilizou o Streamlit para análise e previsão de tumores cerebrais. Isso é essencial para exibir métricas de avaliação de modelos e *insights* de forma clara e compreensível para cientistas de dados e *stakeholders*.

- **Integração com Bibliotecas de *Machine Learning*:** O Streamlit integra-se facilmente com bibliotecas como Scikit-learn, TensorFlow e PyTorch, facilitando a execução de pipelines de ML diretamente na interface, como evidenciado por (Buzea, et al., 2025), que desenvolveram uma aplicação de suporte à decisão clínica com base em *radiomics*.

- **Interface Amigável:** Uma das principais lacunas identificadas nas ferramentas de AutoML é a complexidade das interfaces. O Streamlit resolve essa questão ao oferecer uma interface intuitiva e de fácil interação, permitindo que os cientistas

de dados se concentrem nos resultados e na interpretação, em vez de detalhes técnicos de implementação (Siddiqui & Sorrot, 2024).

- **Capacidade de Personalização:** O Streamlit permite personalizar facilmente fluxos de trabalho e adicionar opções interativas, como seleção de variáveis, ajuste de hiperparâmetros e geração de relatórios a pedido.

2.11.2. Escolha do Streamlit para desenvolvimento da Plataforma

A escolha do Streamlit foi motivada pela necessidade de desenvolver uma solução prática e acessível, que atendesse às necessidades tanto de cientistas de dados experientes quanto de utilizadores com menos conhecimento técnico. A curva de aprendizagem suave, combinada com a robustez para criar aplicações avançadas, posiciona-o como a melhor opção para o desenvolvimento do MLCASE.

2.11.3. Instalação do Streamlit

A instalação do Streamlit pode ser feita de diferentes formas, dependendo do ambiente de desenvolvimento. A forma mais comum é através do terminal, mas há outras opções para quem prefere interfaces gráficas ou ambientes na nuvem (Snowflake Inc., 2025). De acordo com a documentação do Streamlit (Snowflake Inc., 2025), existem quatro formas de instalação:

1. **Instalação via Terminal (método mais comum):** Para instalar o Streamlit rapidamente num ambiente com Python já configurado, basta executar o seguinte comando no terminal:

```
pip install Streamlit
```

Para verificar se a instalação foi concluída com sucesso, pode-se correr o seguinte comando:

```
Streamlit hello
```

Este comando abrirá um exemplo interativo no navegador, demonstrando algumas funcionalidades básicas da biblioteca.

2. **Instalação em ambiente virtual (venv):** Para evitar conflitos entre dependências, pode-se instalar o Streamlit dentro de um ambiente virtual:

```
python -m venv venv # Cria o ambiente virtual
source venv/bin/activate  # Ativa o ambiente virtual para Linux/macOS
venv\Scripts\activate     # Ativa o ambiente virtual para Windows
pip install streamlit # Instala o streamlit
```

3. **Instalação via Anaconda (interface gráfica):** Para quem utiliza o Anaconda, o Streamlit pode ser instalado num ambiente conda:

```
conda create --name streamlit_env python=3.9 # Cria o ambiente virtual
conda activate streamlit_env # Ativa o ambiente virtual
pip install streamlit # Instala o streamlit
```

   - **Streamlit *Community Cloud* ou GitHub *Codespaces*** – Permite correr o Streamlit sem necessidade de instalação local.
   - ***Snowflake*** – Indicado para ambientes corporativos, onde há necessidade de controlo de acesso.

3. Metodologia

A metodologia descrita aqui detalha as etapas e os componentes implementados na plataforma MLCASE, uma solução destinada a automatizar e simplificar o fluxo de trabalho de desenvolvimento de modelos de *Machine Learning* (ML).

3.1. Objetivo Geral

O objetivo deste projeto foi desenvolver uma plataforma que automatizasse diversas etapas do processo de ML, tornando o trabalho dos cientistas de dados mais eficiente e acelerando a criação de modelos. A construção de modelos de ML envolve normalmente várias fases, como o pré-processamento dos dados, a construção e a avaliação dos modelos. Automatizar essas fases permitiu poupar tempo e esforço, possibilitando que os cientistas de dados se dedicassem mais à análise e à interpretação dos resultados.

Com essa automação, procurou-se reduzir a carga de trabalho repetitiva e agilizar o desenvolvimento de modelos, permitindo que os cientistas de dados se concentrassem em tarefas mais complexas, como interpretar os resultados e gerar *insights* significativos.

3.2. Tecnologias e Ferramentas

Esta secção descreve as tecnologias e ferramentas utilizadas no projeto, explicando o seu propósito, funcionalidades principais e o motivo da sua escolha.

3.2.1. *Frameworks* e Bibliotecas de Interface Web

Streamlit

- **Módulos:**
  - **Streamlit**: Permite a criação de interfaces gráficas interativas, com elementos como botões, *sliders*, tabelas interativas e gráficos.

Figura 9. Logotipo Streamlit

  - **streamlit.components.v1:** Possibilita a inclusão de componentes HTML e JavaScript personalizados, permitindo maior flexibilidade e interatividade na interface (Streamlit, 2025).

- **Uso:**
  - Criação de uma interface gráfica intuitiva e acessível para o utilizador.
  - Carregamento de ficheiros (ex.: CSV, Excel).
  - Seleção de colunas e visualização de gráficos gerados pelo modelo.
  - Inclusão de scripts adicionais (ex.: botão para rolar ao topo da página).
- **Motivo da Escolha:**
  - Simplicidade e rapidez na construção de interfaces interativas.
  - Integração com bibliotecas de dados e visualização.
- **Instalação**
  - A aplicação foi desenvolvida num ambiente Python e para a criação da interface, foi instalado o Streamlit diretamente através do terminal, garantindo uma configuração rápida e eficiente.
  - Após o desenvolvimento e testes locais, a aplicação foi carregada online utilizando o Streamlit GitHub *Codespaces*, permitindo a sua execução num ambiente na nuvem, sem necessidade de instalação adicional por parte dos utilizadores. Esta abordagem assegura maior acessibilidade e facilidade de manutenção.
  - O código completo da plataforma pode ser acedido no site GitHub: https://github.com/brunaa0/mlcase-plataform/blob/main/streamlit_app.py e a plataforma acedida por este site: https://mlcase-plataform.streamlit.app/.

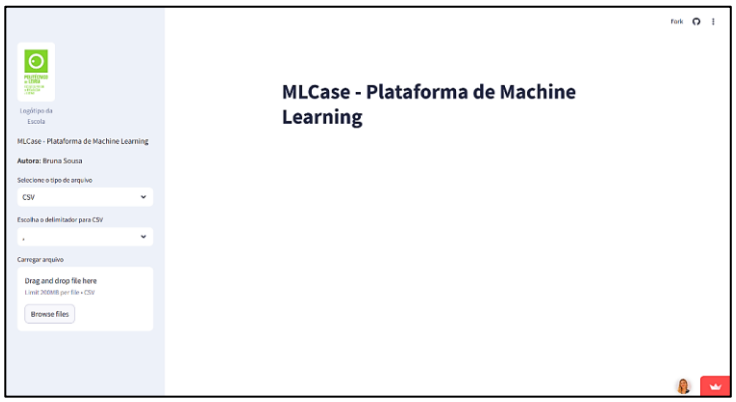


Figura 10. Página Inicial da Plataforma MLCASE

Fonte: Sousa, 2025

3.2.2. Manipulação e Processamento de Dados

Tabela 14. Ferramentas Manipulação e Análise de Dados

Bibliotecas	Uso	Funções Principais	Motivo Escolha	Referência
Pandas	➢ Leitura, manipulação e transformação de dados tabulares. ➢ Operações como filtragem, agregação e conversão de tipos de dados.	➢ Leitura de ficheiros em diferentes formatos. ➢ Manipulação de DataFrames, como seleção de colunas, substituição de valores ausentes e cálculo de estatísticas descritivas. ➢ Configuração de opções de exibição. ➢ Criação de gráficos básicos através de integração com Matplotlib.	➢ Versatilidade para trabalhar com dados estruturados. ➢ Suporte a operações complexas em grandes volumes de dados. ➢ Grande comunidade e documentação robusta.	(Pandas, 2024)
NumPy	➢ Manipulação eficiente de *arrays* multidimensionais. ➢ Cálculo de estatísticas básicas e operações matriciais.	➢ Criação e manipulação de *arrays* multidimensionais. ➢ Funções matemáticas como cálculo de média, desvio padrão e soma cumulativa. ➢ Operações vetorizadas para maior eficiência computacional.	➢ Eficiência no processamento de operações matemáticas intensivas. ➢ Base para várias bibliotecas de *Machine Learning* e visualização.	(NumPy Developers, 2024)

3.2.3. Visualização de Dados

Tabela 15. Bibliotecas de Visualização de Dados

Ferramentas	Uso	Funções Principais	Motivo Escolha	Referência
Matplotlib	➢ Criação de gráficos estáticos para análises exploratórias e relatórios.	➢ Criação de *boxplots* para análise de *outliers*. ➢ Criação de gráficos básicos como gráficos de linhas, barras e dispersão. ➢ Personalização avançada de gráficos (Ex.: cores, títulos, rótulos).	➢ Simplicidade na utilização. ➢ Integração robusta com outras bibliotecas, como Pandas e NumPy. ➢ Grande flexibilidade para personalizar visualizações.	(The Matplotlib development team, 2024)
Seaborn	➢ Criação de visualizações estatísticas apelativas e informativas.	➢ *Heatmaps* para matrizes de correlação. ➢ Gráficos de distribuição. ➢ Visualização de relacionamentos, como gráficos de dispersão com regressão.	➢ Produção de gráficos esteticamente agradáveis com pouco código. ➢ Integração com Pandas para manipulação de *DataFrames*. ➢ Suporte a visualizações estatísticas avançadas.	(Waskom, 2024)
Plotly	➢ Criação de gráficos interativos para análise exploratória detalhada.	➢ Criação de gráficos dinâmicos e responsivos. ➢ Criação de gráficos 3D, mapas e gráficos de séries temporais. ➢ Exportação para relatórios e apresentações em HTML.	➢ Interatividade e flexibilidade nas visualizações. ➢ Suporte a diferentes tipos de gráficos, incluindo 3D. ➢ Integração com *frameworks* web e *notebooks* Jupyter.	(Plotly, 2024)

3.2.4. Algoritmos de *Machine Learning*

Tabela 16. Bibliotecas de Algoritmos de *Machine Learning*

Ferramentas	Uso	Funções Principais	Motivo Escolha	Referência
Scikit-learn	➢ Implementação de algoritmos de aprendizagem automática. ➢ Validação cruzada para avaliação robusta de modelos. ➢ Otimização de hiperparâmetros e métricas de avaliação.	➢ Divisão de Dados em Treino e teste. ➢ Validação Cruzada. ➢ Otimização de hiperparâmetros. ➢ Suporte a modelos. ➢ Métricas de Avaliação.	➢ Ampla gama de algoritmos e ferramentas integradas. ➢ Documentação detalhada e grande suporte da comunidade. ➢ Eficiência computacional e integração com outras bibliotecas como Pandas e NumPy.	(Scikit-learn Developers, 2025)
MLxtend	➢ Suporte à seleção de características com foco em melhorar o desempenho do modelo.	➢ Seleção de características com *SequentialFeatureSelector*. ➢ Integração direta com os modelos do *Scikit-learn.* ➢ Fornece métricas detalhadas sobre a importância das características selecionadas.	➢ Complementa a funcionalidade do *Scikit-learn*. ➢ Ideal para tarefas de seleção de atributos em conjuntos de dados complexos. ➢ Simplicidade e facilidade de uso.	(Raschka, MLxtend, 2023)

3.2.5. Manipulação de Ficheiros

Tabela 17. Bibliotecas de Manipulação de Ficheiros

Ferramentas	Uso	Funções Principais	Motivo Escolha	Referência
IO	➢ Processamento de ficheiros de entrada/saída.	➢ ***BytesIO*:** Permite manipular ficheiros diretamente na memória, sem necessidade de salvar no disco. ➢ Criação de objetos de ficheiros temporários para leitura e escrita dinâmica.	➢ Facilidade em manipular ficheiros de forma eficiente. ➢ Reduz a necessidade de operações no disco, otimizando o desempenho.	(Python Software Foundation, 2025)
Joblib	➢ Serialização de objetos para reutilização em aplicações futuras.	➢ Serialização eficiente de modelos de ML e grandes *arrays* de dados. ➢ Suporte para paralelização e processamento em segundo plano.	➢ Eficiência no armazenamento e recuperação de modelos complexos. ➢ Integração com frameworks como Scikit-learn.	(Joblib developers, 2021)
Pickle	➢ Guardar e carregar objetos *Python* em ficheiros binários.	➢ Conversão de objetos Python para formatos binários. ➢ Recarregamento de objetos serializados.	➢ Solução simples para serializar e de serializar objetos *Python*. ➢ Suporte nativo na biblioteca padrão do *Python*.	(Python Software Foundation, 2025)
Tempfile	➢ Criação e gestão de ficheiros temporários para exportação de relatórios e gráficos.	➢ - ***TemporaryFile*:** Criação de ficheiros temporários descartados após o uso. ➢ ***NamedTemporaryFile*:** Criação de ficheiros temporários com nomes acessíveis no sistema de ficheiros.	➢ Simplificação da gestão de ficheiros temporários. ➢ Utilização segura e descartável sem necessidade de limpeza manual.	(Python Software Foundation, 2025)

3.2.6. Criação de Relatórios PDF

Tabela 18. Bibliotecas de Criação de Relatórios PDF

Ferramentas	Uso	Funções Principais	Motivo Escolha	Referência
FPDF	➢ Criação de relatórios PDF personalizados com estatísticas, tabelas e gráficos.	➢ Criação de documentos PDF do zero. ➢ Inserção de textos, gráficos, tabelas e imagens. ➢ **Configuração de Layouts:** Personalização de margens, fontes e alinhamentos. ➢ Criação de relatórios com texto em diferentes idiomas, incluindo caracteres especiais.	➢ Ferramenta leve e fácil de usar. ➢ Totalmente escrita em Python, sem dependências externas complexas. ➢ Suporte robusto para criação de relatórios personalizáveis.	(GitHub, 2025)

3.2.7. Operações Estatísticas e Matemáticas

Tabela 19. Bibliotecas de Operações Estatísticas e Matemáticas

Ferramentas	Uso	Funções Principais	Motivo Escolha	Referência
SciPy	➢ Realização de cálculos estatísticos avançados. ➢ Manipulação de matrizes, incluindo matrizes esparsas.	➢ Representação eficiente de dados em matrizes esparsas, ideal para lidar com grandes volumes de dados onde a maioria dos elementos é zero ou ausente. ➢ Ferramentas para cálculo de estatísticas descritivas e testes de hipóteses.	➢ Suporte robusto para cálculos avançados. ➢ Eficiência no tratamento de matrizes grandes e esparsas. ➢ Extensa documentação e integração com NumPy.	(The SciPy community, 2025)
Decimal	➢ Trabalhar com números de alta precisão, especialmente úteis para cálculos financeiros e científicos.	➢ Permite definir o número de casas decimais para evitar erros de arredondamento. ➢ **Operações Matemáticas Seguras:** Soma, subtração, multiplicação e divisão com precisão.	➢ Adequado para cenários que exigem cálculos precisos. ➢ Evita erros de precisão comuns em números de ponto flutuante.	(Python Software Foundation, 2025)
Fractions	➢ Representação de números como frações, preservando a precisão de valores racionais.	➢ **Operações Matemáticas com Frações:** Soma, subtração, multiplicação e divisão de frações. ➢ **Conversão entre Tipos:** Transformação de números inteiros, decimais ou *strings* em frações exatas.	➢ Ideal para trabalhar com valores racionais em que a precisão é crítica. ➢ Facilita o entendimento e manipulação de valores fracionários.	(Python Software Foundation, 2025)

3.2.8. Outros Utilitários

Tabela 20. Outras Bibliotecas

Ferramentas	Uso	Funções Principais	Motivo Escolha	Referência
Requests	➢ Realização de requisições HTTP para interagir com *APIs* e realizar *downloads* de ficheiros ou imagens.	➢ ***get*:** Requisita dados de *URLs* específicas. ➢ ***post*:** Envia dados para servidores. ➢ ***download*:** Obtenção de ficheiros diretamente da web.	➢ Simplicidade e eficiência na interação com *APIs*. ➢ Ferramenta amplamente utilizada e bem documentada.	(A Kenneth Reitz Project, s.d.)
Datetime	➢ Manipulação e formatação de datas e horas.	➢ Criação de Datas/Horas. ➢ **Operações com Datas:** Soma ou subtração de dias/horas. ➢ **Formatação:** Conversão de objetos *datetime* para *strings* formatadas.	➢ Ferramenta robusta para manipulação de datas. ➢ Parte da biblioteca padrão do *Python*, sem dependências adicionais.	(Python Software Foundation, 2025)
OS	➢ Execução de operações no sistema de ficheiros e ambiente do sistema operacional.	➢ Gestão de Diretórios. ➢ Obtenção de Informações do Ambiente. ➢ Gestão de Caminhos.	➢ Permite interação direta com o sistema operacional. ➢ Essencial para manipulação de ficheiros e diretórios.	(Python Software Foundation, 2025)
Time	➢ - Medição de tempos de execução e introdução de atrasos.	➢ **Medição de Tempo:** *time.time* para marcar início e fim de processos. ➢ **Atrasos:** *time.sleep* para introduzir pausas no código.	➢ Ferramenta simples para cronometrar processos. ➢ Útil para otimizar e controlar execuções.	(Python Software Foundation, 2025)

Re (Expressões Regulares)	➢ Processamento avançado de *strings* com validação de padrões.	➢ Validação de Padrões. ➢ Substituição e Divisão. ➢ **Compilação de Padrões**: Criação de padrões reutilizáveis.	➢ Ferramenta poderosa para manipulação de *strings.* ➢ Ideal para tarefas como limpeza e validação de dados.	(Python Software Foundation, 2025)

3.3. Arquitetura Geral

A plataforma foi desenvolvida utilizando o *framework* Streamlit para construção de interfaces interativas e simples, integrando diversas bibliotecas para manipulação de dados, visualização e desenvolvimento de modelos de *Machine Learning*. O fluxo de trabalho é modular, sendo composto pelas seguintes etapas principais, cada uma destas descrita em detalhe no ponto seguinte.

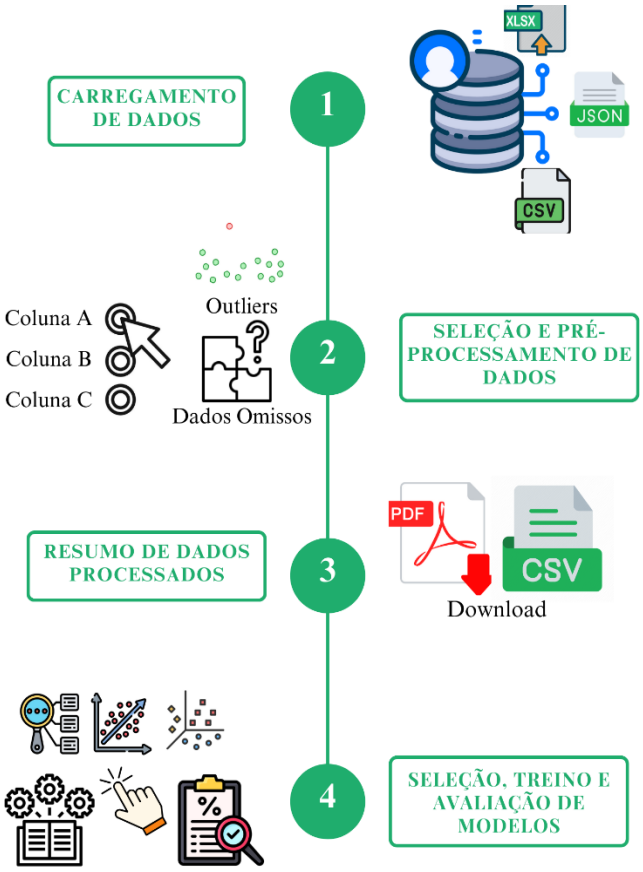


Figura 11. Fluxo de Etapas Principais

3.3.1. Carregamento de dados

O objetivo da etapa de carregamento de dados é permitir que o utilizador carregue o conjunto de dados em formatos variados, como CSV, Excel ou JSON. A plataforma lida com os diferentes formatos e prepara os dados para as etapas subsequentes.

- **Função de *upload*:** O código começa com a função *upload_file(),* que apresenta ao utilizador uma interface para escolher o tipo de arquivo que deseja carregar. Dependendo do formato escolhido, o código usa uma função auxiliar (*load_data()*) para carregar os dados de acordo com o tipo selecionado.

    - **CSV:** Se o tipo de arquivo for CSV, o código chama a função *choose_delimiter(),* que permite ao utilizador escolher o delimitador (vírgula, ponto e vírgula, tabulação, barra vertical, ou até mesmo um delimitador personalizado).

    - **Excel:** Para arquivos Excel, é usado *pd.read_excel().*

    - **JSON:** Para arquivos JSON, a função *pd.read_json()* é usada.

- Após o upload do ficheiro é necessário carregar no botão de dados carregados para seguir para o passo seguinte.

3.3.2. Seleção e pré-processamento de dados

Após o carregamento, a função *initialize_state()* inicializa as variáveis de estado na sessão do Streamlit, como as colunas presentes no ficheiro e os tipos de dados. Isso é feito para facilitar o controlo do processo de pré-processamento de dados nas etapas seguintes. Esta etapa está dividida em 3 subetapas sendo elas:

- Seleção, Identificação e Discretização de variáveis.
- Tratamento de Valores Ausentes.
- Deteção de *Outliers.*

Seleção, Identificação e Discretização de variáveis

Seleção de Colunas: O código exibe uma visualização preliminar dos dados e permite que o utilizador selecione as colunas para trabalhar. Isso é feito usando o método *multiselect()* do Streamlit, permitindo que o utilizador escolha múltiplas colunas do conjunto de dados. Por defeito, estão todas as colunas selecionas. Caso o utilizador deseje remover alguma coluna, basta clicar no ícone de remoção (representado por um "X"). Se, por algum motivo, o utilizador remover uma coluna acidentalmente, ele pode clicar na seta para baixo para aceder a lista de colunas removidas. Ao clicar sobre a coluna desejada, ela será adicionada novamente ao conjunto de dados.

Identificação de Tipos de Variáveis: O código permite ao utilizador selecionar o tipo de variável para cada coluna selecionada. As opções disponíveis são as seguintes:

- **Numérica:** Se a coluna for identificada como numérica, o sistema abre uma caixa adicional para que o utilizador escolha o tipo específico de dado numérico:
  - *Int*
  - *Float*
  - *Dec*
  - *Complex*
  - *Frac*
  - *Bool*

- **Categórica:** Para colunas que contêm categorias ou rótulos (por exemplo, *strings*).
- **Data:** Para colunas que contêm valores temporais (datas ou horas).

O código deteta automaticamente o tipo de dados de cada coluna e preenche o tipo de variável correspondente. No entanto, o utilizador tem a possibilidade de modificar o tipo de variável sugerido, caso necessário.

Discretização de Variáveis: Caso o utilizador opte por discretizar uma variável numérica, o código permite a conversão dos valores contínuos em categorias discretas, por meio de "*bins*" (intervalos). O processo de discretização é feito utilizando a função *pd.cut()* do pandas, que converte uma variável numérica contínua em uma variável categórica, baseada nos intervalos definidos pelo utilizador. Antes de realizar a discretização, a plataforma apresenta um diagnóstico dos dados, fornecendo informações como Valor Mínimo, Valor Máximo**,** Média, Mediana e Número de Valores Ausentes.

Essas informações ajudam o utilizador a entender melhor a distribuição dos dados antes de aplicar a discretização. Este processo de discretização ajuda a transformar variáveis contínuas em categorias, o que pode ser útil para análise de dados categóricos ou para ML que necessitam de variáveis discretas.

O código deteta automaticamente os intervalos e rótulos de cada coluna e preenche correspondente. No entanto, o utilizador tem a possibilidade de modificar o sugerido, caso necessário.

- **Passos no Processo de Discretização:**
    - **Definição dos Intervalos (*Bins*):** O utilizador deve especificar os intervalos para os dados, fornecendo uma lista de valores numéricos separados por vírgulas. Estes valores representam os limites dos "*bins*", ou intervalos, nos quais os dados serão classificados.

    - **Definição dos *Labels* (Rótulos):** Após definir os intervalos, o utilizador deve fornecer rótulos para as categorias geradas a partir dos *bins*. Esses rótulos são usados para identificar as categorias geradas após a discretização.
    - **Confirmação:** Após fornecer os intervalos e os rótulos, o utilizador deve confirmar a discretização. Ao clicar no botão "Confirmar Discretização para [variável]", a variável é convertida nos intervalos e rótulos definidos, e os valores da coluna são substituídos pelos rótulos correspondentes aos *bins*.

Tratamento de Valores Ausentes

O código permite ao utilizador selecionar a abordagem para lidar com valores ausentes nas colunas selecionadas. A plataforma oferece diferentes opções de tratamento, dependendo do tipo de variável (Numérica ou Categórica). O tratamento é realizado utilizando funções como *fillna()* para preenchimento e *dropna()* para exclusão de dados ausentes.

- **Deteção de Valores Ausentes**

O código identifica automaticamente as colunas que contêm valores ausentes. Para cada coluna, o número de valores ausentes é calculado utilizando o método *isna().sum()* do pandas. A plataforma exibe uma tabela resumo, na qual são apresentados o número de valores ausentes para cada coluna e os respetivos nomes das colunas que contêm esses valores.

- **Tratamento de Valores Ausentes**

O código permite ao utilizador escolher entre diferentes métodos de tratamento para os valores ausentes. As opções de tratamento incluem:

  - **Substituição por Média:** Para colunas numéricas, o código pode substituir os valores ausentes pela média da coluna. Isso é feito com a função *fillna*(), utilizando o valor retornado por *mean*() para preencher as lacunas.

  - **Substituição por Mediana:** O utilizador também pode optar por substituir os valores ausentes pela mediana da coluna, utilizando *fillna*() e a função *median*().
  - **Substituição por Moda:** Outra opção disponível é substituir os valores ausentes pela moda (o valor mais frequente) da coluna. Para isso, é usado o método *mode*().*iloc*[0], que retorna o valor mais frequente da coluna.

  - **Exclusão de Linhas com Valores Ausentes:** O código também oferece a opção de excluir as linhas que contêm valores ausentes, utilizando o método *dropna*() do pandas. Isso pode ser útil quando a quantidade de dados ausentes é pequena e a exclusão não afetará significativamente a análise.

  - **Substituição por Valor Constante:** O utilizador pode fornecer um valor constante para substituir os valores ausentes, permitindo um controlo total sobre o preenchimento de dados ausentes.

- **Manter Valores Ausentes:** Se o utilizador desejar, os valores ausentes podem ser mantidos sem qualquer alteração. Isso pode ser útil em casos específicos onde os valores ausentes têm um significado especial ou a análise futura requer que esses valores ausentes sejam preservados.

- **Implementação no Código**

A função *apply_missing_value_treatment*() é responsável por aplicar as mudanças. O método de tratamento escolhido é passado como parâmetro, e a coluna correspondente é tratada de acordo. Por exemplo, se o método for "Substituir por Média", a função irá preencher os valores ausentes com a média da coluna.

Além disso, o código sugere automaticamente o melhor método de tratamento de valores ausentes com base na percentagem de valores ausentes na coluna. Para colunas numéricas, se mais de 50% dos valores estiverem ausentes, o código sugere a exclusão da coluna. Para colunas com menos de 50% de valores ausentes, ele sugere o preenchimento com a mediana.

- **Exibição para o Utilizador**

O utilizador tem controlo total sobre o método de tratamento a ser aplicado. Após selecionar o método desejado, o código aplica a modificação e mostra o número de valores ausentes antes e depois do tratamento.

Deteção de *Outliers*

O código da plataforma deteta e trata os *outliers* nas variáveis numéricas de forma automatizada ou com a possibilidade de personalização por parte do utilizador. A plataforma oferece diferentes abordagens para detetar e lidar com esses valores.

- **Passos no Tratamento de *Outliers*:**
  - **Identificação dos *Outliers***: O código utiliza métodos estatísticos para identificar os *outliers* nas colunas numéricas.
    - **IQR (Intervalo Interquartil):** O código calcula o intervalo interquartil (IQR), que é a diferença entre o primeiro quartil (Q1)

e o terceiro quartil (Q3) dos dados. Qualquer valor fora da faixa definida como [Q1 - 1.5 * IQR, Q3 + 1.5 * IQR] é considerado um *outlier*. Esta técnica é amplamente usada para detetar valores extremamente discrepantes.

- **Desvio Padrão**: Outra técnica possível é a utilização do desvio padrão. Valores que estão a uma distância superior a 2 ou 3 desvios padrão da média são considerados *outliers*.

- **Exibição de *Outliers*:** Quando detetados, os *outliers* são apresentados ao utilizador numa tabela, onde são exibidos o número total de *outliers*, o número de outliers severos e é apresentado também um *boxplot* inicial.
- **Exibição de Dados para Tomada de Decisão:** Após a exibição do *boxplot* inicial e da tabela de resumo, a plataforma fornece detalhes adicionais sobre a coluna selecionada. Essas informações incluem:
  - Total de elementos na coluna.
  - Número e percentual de *outliers.*
  - Número de *outliers* severos.
  - Assimetria (*Skewness*): Uma medida de simetria dos dados, que pode ajudar a decidir se a distribuição dos dados justifica a remoção ou a modificação dos *outliers*.

- **Tratamento de *Outliers*:** O utilizador pode escolher entre diferentes métodos para tratar os *outliers*, dependendo da estratégia desejada. As opções disponíveis são:
  - **Remoção de *Outliers*:** O código pode excluir todas as linhas que contêm *outliers*, utilizando a função *drop()* do pandas, removendo diretamente os valores discrepantes do conjunto de dados.
  - **Remoção de *Outliers* Severos:** Esta opção remove apenas os *outliers* que foram considerados severos, ou seja, aqueles que estão fora de limites mais amplos, definidos pelo IQR ou desvio padrão.
  - **Substituição por Média ou Mediana:** Os *outliers* podem ser substituídos por valores mais representativos da distribuição dos dados, como a média ou a mediana da coluna. A substituição é feita

utilizando a função *fillna()* ou manipulando diretamente os valores das células com *outliers*.

- **Substituir por Limites:** O código pode "cortar" os *outliers*, substituindo-os por um valor máximo ou mínimo dentro de um intervalo mais razoável, definido pelo utilizador ou pelos limites calculados, como o percentil 95.
- **Sem Ação:** Esta opção não remove nem substitui nenhum dos *outliers*, mantendo os dados intactos.

- **Aplicação do Tratamento:** Após a escolha do método de tratamento, o código aplica a modificação ao conjunto de dados automaticamente. A plataforma também permite que o utilizador visualize o impacto da remoção ou substituição dos *outliers*, comparando os dados antes e depois do tratamento.

3.3.3. Resumo de dados Processados

A plataforma oferece uma visualização detalhada dos dados carregados, permitindo ao utilizador obter um resumo das principais características do conjunto de dados. O código foi desenvolvido para gerar uma tabela de estatísticas descritivas e exibir gráficos que facilitam a análise inicial do *dataset*.

- **Estatísticas Descritivas:** A primeira etapa do processo de resumo de dados consiste no cálculo e exibição de informações descritivas essenciais para cada coluna numérica e categórica do *dataset*. As estatísticas descritivas são calculadas utilizando a função *describe*() do pandas, que fornece os seguintes valores para cada coluna:
  - Número de linhas e colunas.
  - Média.
  - Desvio padrão.
  - Valores mínimo e máximo.
  - Percentis de 25%, 50% e 75%.

Este resumo é exibido numa tabela interativa, onde o utilizador pode selecionar as colunas desejadas para visualização.

- **Resumo de Valores Ausentes:** A plataforma identifica e informa o utilizador sobre a presença de valores ausentes, conforme o tratamento realizado nas etapas anteriores. Caso não existam valores ausentes, será exibida a mensagem "Não há valores ausentes nas variáveis selecionadas". Se valores ausentes forem encontrados, a plataforma apresenta a quantidade de dados ausentes em cada coluna.

- **Resumo de *Outliers*:** A plataforma também verifica a presença de *outliers* nas variáveis selecionadas, conforme os tratamentos realizados anteriormente. Caso não existam *outliers*, será exibida a mensagem "Não há *outliers* nas variáveis selecionadas." Se *outliers* forem detetados, o utilizador será informado sobre a quantidade de dados afetados por *outliers* em cada coluna.

- ***Boxplot* das Variáveis Numéricas:** A plataforma gera um *boxplot* para as variáveis numéricas selecionadas, permitindo ao utilizador observar a distribuição dos dados e identificar visualmente os *outliers*, caso existam.
- **Matriz de Correlação das Variáveis:** Uma matriz de correlação entre as variáveis é exibida, permitindo que o utilizador identifique possíveis relações entre as colunas do *dataset*.

- **Botão para download do Resumo em PDF:** O utilizador pode baixar um PDF contendo o resumo das estatísticas descritivas, o diagnóstico de valores ausentes, a análise de *outliers* e a matriz de correlação, para consulta ou partilha.

- **Botão para download do *Dataset* Tratado:** O utilizador pode também baixar o *dataset* tratado, que inclui todas as alterações realizadas, como a remoção ou substituição de valores ausentes e *outliers*.

3.3.4. Seleção, Treino e Avaliação de Modelos

Após a preparação do *dataset*, segue-se a parte principal do desenvolvimento da plataforma. Esta etapa esta está dividida em 3 subetapas sendo ela:

- Modelos de Classificação.
- Modelos de Regressão.
- Modelos de *Clustering*.

Modelos de Classificação

Esta fase da plataforma é dedicada à seleção e treino de modelos de classificação. Ela está organizada em etapas sequenciais que guiam o utilizador desde a escolha do tipo de modelo até a avaliação final. A seguir estão os passos detalhados que o utilizador deve seguir:

1. **Escolher o Modelo:** O primeiro passo é escolher o tipo de modelo que será utilizado. A plataforma oferece três tipos principais de modelos de classificação:
    - *Support Vector Classification* (SVC).
    - *K-Nearest Neighbors* (KNN).
    - *Random Forest.*
2. **Escolher a Coluna Alvo:** O utilizador deve selecionar qual coluna do conjunto de dados será usada como coluna alvo (variável dependente). Esta coluna será a variável que o modelo tentará classificar.
3. **Uso de *Grid Search*:** O próximo passo envolve a escolha de *Grid Search* para otimização dos parâmetros do modelo. A plataforma oferece duas opções:
    - **Deixar a plataforma escolher automaticamente os melhores parâmetros**: Neste caso, o algoritmo vai automaticamente testar várias combinações de parâmetros e escolher a melhor.
    - **Escolher manualmente os parâmetros de *Grid Search***: O utilizador pode optar por definir manualmente os valores dos parâmetros para otimização. Se esta opção for escolhida, o utilizador deve selecionar os parâmetros adequados para cada modelo de classificação escolhido. Abaixo estão os parâmetros a definir para cada modelo:

- **KNN**
  - ***n_neighbors***: Número de vizinhos a serem considerados para a classificação.
  - ***weights***: Define a forma como os vizinhos influenciam a previsão. Pode ser "uniforme" (todos os vizinhos têm o mesmo peso) ou "distância" (os vizinhos mais próximos têm maior peso).
- **SVC:**
  - **C**: Parâmetro de regularização que controla o *trade-off* entre maximizar a margem e minimizar o erro de classificação.
  - ***kernel***: Pode ser linear ou rbf (*Radial Basis Function*). Se o *kernel* escolhido for rbf, é necessário definir o parâmetro *gamma*, que controla a largura do *kernel*.
    - ***gamma***: Parâmetro que controla a forma do *kernel* RBF. Valores mais altos fazem com que o modelo se ajuste mais aos dados de treino, enquanto valores mais baixos fazem o modelo ser mais generalista.
- **RandomForest:**
  - **n_*estimators***: Número de árvores a serem usadas no modelo. Mais árvores geralmente melhoram o desempenho do modelo, mas aumentam o tempo de treino.

4. **Escolher os Parâmetros de Validação:** A próxima etapa é escolher como os dados serão validados durante o treino. Existem duas opções de validação:
   - **Divisão de Treino e Teste**: O conjunto de dados será dividido em duas partes, uma para treino e outra para teste. O utilizador pode definir, através de um *slider*, a proporção da divisão.

- ***Holdout***: O utilizador pode optar por uma abordagem de *holdout*, onde o treino será feito numa parte do *dataset* e o teste outra. O utilizador pode definir, através de um *slider*, a proporção da divisão.

5. **Treinar o Modelo:** Após definir todos os parâmetros o utilizador deve clicar no botão "Treinar Modelo". A plataforma treina o modelo utilizando as configurações escolhidas e apresenta as métricas de desempenho correspondentes.

Após clicar no botão "Treinar Modelo", a plataforma exibe um Resumo das Escolhas Feitas, incluindo os detalhes do modelo selecionado, os parâmetros escolhidos e as configurações de validação. Depois é exibida uma tabela com as métricas do modelo treinado permitindo ao utilizador avaliar o desempenho do modelo e um gráfico de desempenho (gráfico de barras com as métricas). Após visualizar os resultados, o utilizador pode clicar no botão "Avançar para Seleção de *Features*" para prosseguir com a próxima etapa.

6. **Seleção de *Features*:** Na fase de Seleção de *Features*, o utilizador deve:

   - **Escolher o Critério de Avaliação**: A plataforma permite ao utilizador selecionar o critério de *scoring*, que define a métrica a ser utilizada para a avaliação das *features*. As opções incluem *Accuracy*, *Precision*, *Recall* ou F1-*Score*.

   - **Escolher o Método de Seleção de *Features***: A plataforma oferece dois métodos para seleção das *features*:

     - **Seleção Automática**: A plataforma escolhe automaticamente o melhor número de *features* com base no critério de avaliação selecionado.

     - **Seleção Manual**: O utilizador pode escolher manualmente o número de *features* a ser considerado no modelo. O número de *features* pode ser selecionado através de um *slider* (barra deslizante), onde o utilizador ajusta a quantidade de *features* conforme a necessidade.

   - **Confirmar a Seleção de *Features***: Após selecionar o método e definir o número de *features*, o utilizador deve clicar no botão " Treinar Modelo com

Features Selecionadas " para que a plataforma aplique a seleção e avance para a etapa seguinte.

7. **Treino do Modelo com *Features* Selecionadas:** Após o utilizador clicar no botão "Treinar Modelo com *Features* Selecionadas", é apresentada uma tabela com as métricas obtidas no treino com a seleção de *features*. Depois o utilizador deve carregar no botão de "Comparar Modelos" para seguir para a próxima etapa onde será possível comparar as métricas dos modelos.

8. **Comparação dos Resultados do Treino dos Modelos:** Após o utilizador clicar no botão "Comparar Modelos", a plataforma apresenta uma comparação dos resultados obtidos nos modelos de classificação, antes e após a seleção de *features*. Nesta etapa é mostrado:

    - **Tabela de Comparação dos Resultados:** É exibida uma tabela comparativa contendo as métricas de desempenho e os melhores parâmetros, caso o utilizador tenha selecionado usar *Grid Search*, para os dois cenários:

        - **Sem Seleção de *Features*:** Resultados do modelo treinado com todas as *features*, sem realizar a seleção de características.

        - **Com Seleção de *Features*:** Resultados do modelo treinado com as *features* selecionadas, após o processo de escolha.

    - **Gráfico de comparação de métricas:** É apresentado um gráfico de comparação dos treinos realizados (antes e após seleção de *features*) com a métrica que o utilizador selecionou no *scoring* .

    - **Melhor Modelo:** É indicado o melhor modelo, consoante a métrica escolhida pelo utilizador no *scoring*.

9. **Resumo Final dos Modelos Treinados:** Após o utilizador clicar no botão "Seguir para Resumo Final", a plataforma exibe um resumo final das configurações e resultados dos modelos treinados.

- **Configurações Utilizadas:** A plataforma apresenta uma seção de Configurações Utilizadas, que inclui o tipo de modelo que foi escolhido para treino e o modelo específico que foi utilizado.

- **Informações dos Conjuntos de Dados:** A plataforma mostra o número de amostras de treino, o número de amostras de teste, o número de *features* originais, o número de *features* após seleção e quais as *features* que foram selecionadas.

- **Comparação de Métricas:** Em seguida, é apresentada uma tabela de Comparação de Métricas que contêm as métricas de desempenho e os melhores parâmetros, caso o utilizador tenha selecionado usar *Grid Search*, para os dois cenários:

    - **Sem Seleção de *Features***: Resultados do modelo treinado com todas as *features*.

    - **Com Seleção de *Features***: Resultados do modelo após a seleção das melhores *features*.

- **Gráfico de Comparação de Métricas:** A plataforma também exibe um gráfico interativo para comparar as métricas em ambos os cenários. O utilizador pode escolher a métrica que deseja visualizar no gráfico, permitindo uma comparação visual clara entre os modelos com e sem seleção de *features*.

- **Indicação do melhor modelo:** A plataforma fornece a indicação do melhor modelo treinado com base na métrica que o utilizador escolheu no passo de seleção de *features.*

- **Interpretação das Métricas:** Após as comparações, a plataforma fornece uma interpretação das métricas, explicando o desempenho dos modelos. A conclusão geral do desempenho do modelo é fornecida, sugerindo possíveis melhorias, como ajustes nos hiperparâmetros ou aprimoramento dos dados de treino.

- ***Download* do Melhor Modelo Treinado:** Após analisar os resultados, o utilizador pode fazer o *download* do melhor modelo treinado. A plataforma permite:

    - ***Download* do Melhor Modelo**: O modelo treinado é salvo com um nome de arquivo, como por exemplo ***best_model_com_selecao_features*.pkl**, para que o utilizador possa carregá-lo novamente no futuro sem precisar treiná-lo novamente.

    - ***Download* do Relatório PDF**: O utilizador pode fazer o *download* do relatório em PDF contendo todas as métricas, interpretações e comparações feitas durante o treino, para consulta ou apresentação.

Após o *download,* o utilizador pode clicar em "Concluir" para finalizar o processo de treino de modelos.

Modelos de Regressão

Esta fase da plataforma é dedicada à seleção e treino de modelos de regressão. Ela está organizada em etapas sequenciais que guiam o utilizador desde a escolha do tipo de modelo até a avaliação final. A seguir estão os passos detalhados que o utilizador deve seguir:

1. **Escolher o Modelo:** O primeiro passo é escolher o tipo de modelo de regressão que será utilizado. A plataforma oferece dois tipos principais de modelos de regressão:

    - Regressão Linear Simples (RLS).

    - Regressão por Vetores de Suporte (SVR).

2. **Escolher a Coluna Alvo:** O utilizador deve selecionar qual coluna do conjunto de dados será usada como coluna alvo (variável dependente). Esta coluna será a variável que o modelo tentará prever.

3. **Uso de *Grid Search*:** O próximo passo envolve a escolha de *Grid Search* para otimização dos parâmetros do modelo. A plataforma oferece duas opções:

- **Deixar a plataforma escolher automaticamente os melhores parâmetros**: Neste caso, o algoritmo vai automaticamente testar várias combinações de parâmetros e escolher a melhor.

- **Escolher manualmente os parâmetros de *Grid Search***: O utilizador pode optar por definir manualmente os valores dos parâmetros para otimização. Se esta opção for escolhida, o utilizador deve selecionar os parâmetros adequados para cada modelo de classificação escolhido. Abaixo estão os parâmetros a definir para cada modelo:

    - **SVR:**

        - **C**: Parâmetro de regularização que controla o *trade-off* entre maximizar a margem e minimizar o erro de classificação.

        - ***epsilon***: Define a margem dentro da qual não ocorrem penalidades.

        - ***kernel***: Pode ser linear ou rbf (*Radial Basis Function*).

    - **RLS:** Não há parâmetros adicionais, pois é um modelo linear simples que depende diretamente das variáveis independentes.

4. **Escolher os Parâmetros de Validação:** A próxima etapa é escolher como os dados serão validados durante o treino. Existem duas opções de validação:

    - **Divisão de Treino e Teste**: O conjunto de dados será dividido em duas partes, uma para treino e outra para teste. O utilizador pode definir, através de um *slider*, a proporção da divisão.

    - ***Holdout***: O utilizador pode optar por uma abordagem de *holdout*, onde o treino será feito numa parte do *dataset* e o teste outra. O utilizador pode definir, através de um *slider*, a proporção da divisão.

5. **Treinar o Modelo:** Após definir todos os parâmetros o utilizador deve clicar no botão "Treinar Modelo". A plataforma treina o modelo utilizando as configurações escolhidas e apresenta as métricas de desempenho correspondentes.

Após clicar no botão "Treinar Modelo", a plataforma exibe um Resumo das Escolhas Feitas, incluindo os detalhes do modelo selecionado, os parâmetros escolhidos e as configurações de validação. Depois é exibida uma tabela com as métricas do modelo treinado permitindo ao utilizador avaliar o desempenho do modelo e um gráfico de desempenho (gráfico de barras com as métricas). Após visualizar os resultados, o utilizador pode clicar no botão "Avançar para Seleção de *Features*" para prosseguir com a próxima etapa.

6. **Seleção de *Features*:** Na fase de Seleção de *Features*, o utilizador deve:

    - **Escolher o Critério de Avaliação**: A plataforma permite ao utilizador selecionar o critério de *scoring*, que define a métrica a ser utilizada para a avaliação das *features*. As opções incluem R^2, MAE e MSE.

    - **Escolher o Método de Seleção de *Features***: A plataforma oferece dois métodos para seleção das *features*:

        - **Seleção Automática**: A plataforma escolhe automaticamente o melhor número de *features* com base no critério de avaliação selecionado.

        - **Seleção Manual**: O utilizador pode escolher manualmente o número de *features* a ser considerado no modelo. O número de *features* pode ser selecionado através de um *slider* (barra deslizante), onde o utilizador ajusta a quantidade de *features* conforme a necessidade.

    - **Confirmar a Seleção de *Features***: Após selecionar o método e definir o número de *features*, o utilizador deve clicar no botão "Confirmar Seleção de *Features*" para que a plataforma aplique a seleção e avance para a etapa seguinte.

7. **Treino do Modelo com *Features* Selecionadas:** Após o utilizador clicar no botão "Treinar Modelo com *Features* Selecionadas", é apresentada uma tabela com as métricas obtidas no treino com a seleção de features. Depois o utilizador deve carregar no botão de “Comparar Modelos” para seguir para a próxima etapa onde será possível comparar as métricas dos modelos.

8. **Comparação dos Resultados do Treino dos Modelos:** Após o utilizador clicar no botão " Comparar Modelos ", a plataforma apresenta uma comparação dos resultados obtidos nos modelos de classificação, antes e após a seleção de *features*. Nesta etapa é mostrado:

    - **Tabela de Comparação dos Resultados:** É exibida uma tabela comparativa contendo as métricas de desempenho e os melhores parâmetros, caso o utilizador tenha selecionado usar *Grid Search*, para os dois cenários:

        - **Sem Seleção de *Features*:** Resultados do modelo treinado com todas as *features*, sem realizar a seleção de características.

        - **Com Seleção de *Features*:** Resultados do modelo treinado com as *features* selecionadas, após o processo de escolha.

    - **Gráfico de comparação de métricas:** É apresentado um gráfico de comparação dos treinos realizados (antes e após seleção de *features*) com a métrica que o utilizador selecionou no *scoring* .

    - **Melhor Modelo:** É indicado o melhor modelo, consoante a métrica escolhida pelo utilizador no *scoring*.

9. **Resumo Final dos Modelos Treinados:** Após o utilizador clicar no botão "Seguir para Resumo Final", a plataforma exibe um resumo final das configurações e resultados dos modelos treinados.

    - **Configurações Utilizadas:** A plataforma apresenta uma seção de configurações utilizadas, que inclui o tipo de modelo que foi escolhido para treino e o modelo específico que foi utilizado.

    - **Informações dos Conjuntos de Dados:** A plataforma mostra o número de amostras de treino, o número de amostras de teste, o número de *features* originais, o número de *features* após seleção e quais as *features* que foram selecionadas.

    - **Comparação de Métricas:** Em seguida, é apresentada uma tabela de Comparação de Métricas que contêm as métricas de desempenho e os

melhores parâmetros, caso o utilizador tenha selecionado usar *Grid Search,* para os dois cenários:

  - **Sem Seleção de *Features***: Resultados do modelo treinado com todas as *features*.

  - **Com Seleção de *Features***: Resultados do modelo após a seleção das melhores *features*.

- **Gráfico de Comparação de Métricas:** A plataforma também exibe um gráfico interativo para comparar as métricas em ambos os cenários. O utilizador pode escolher a métrica que deseja visualizar no gráfico, permitindo uma comparação visual clara entre os modelos com e sem seleção de *features*.

- **Indicação do melhor modelo:** A plataforma fornece a indicação do melhor modelo treinado com base na métrica que o utilizador escolheu no passo de seleção de *features.*

- **Interpretação das Métricas:** Após as comparações, a plataforma fornece uma interpretação das métricas, explicando o desempenho dos modelos. A conclusão geral do desempenho do modelo é fornecida, sugerindo possíveis melhorias, como ajustes nos hiperparâmetros ou aprimoramento dos dados de treino.

- ***Download* do Melhor Modelo Treinado:** Após analisar os resultados, o utilizador pode fazer o *download* do melhor modelo treinado. A plataforma permite:

  - ***Download* do Melhor Modelo**: O modelo treinado é salvo com um nome de arquivo, como por exemplo ***best_model_com_selecao_features*.pkl**, para que o utilizador possa carregá-lo novamente no futuro sem precisar treiná-lo novamente.

  - ***Download* do Relatório PDF**: O utilizador pode fazer o *download* do relatório em PDF contendo todas as métricas, interpretações e comparações feitas durante o treino, para consulta ou apresentação.

Após o *download*, o utilizador pode clicar em "Concluir" para finalizar o processo de treino de modelos.

Modelos de *Clustering*

Esta fase da plataforma é dedicada à seleção e treino de modelos de *clustering*. Ela está organizada em etapas sequenciais que guiam o utilizador desde a escolha do modelo até à avaliação final. Seguem os passos detalhados que o utilizador deve seguir:

1. **Escolher o Modelo:** O primeiro passo é escolher o tipo de modelo de *clustering* que será utilizado. A plataforma oferece dois modelos de principais:
    - K-*Means*.
    - *Clustering* Hierárquico.
2. **Escolher a Coluna Alvo:** Ao contrário dos modelos de classificação e regressão, o *clustering* não exige uma coluna alvo.
3. **Redução de Dimensionalidade com PCA:** A p**l**ataforma permite determinar automaticamente ou manualmente o nº de componentes. Por defeito está selecionado automaticamente. Para escolher manualmente basta tirar o pisco da opção. Depois de confirmar a configuração do PCA é possível visualizar os dados após o PCA.
4. **Determinação Número de Clusters Ideal**: A determinação do número de *clusters* é um passo crucial no processo de *clustering*. A plataforma oferece duas abordagens para definir o número de *clusters*:
    - **Automática**: A plataforma determina automaticamente o número ideal de *clusters*, com base na métrica *Silhouette Score*.
    - **Manual**: O utilizador pode optar por definir manualmente o número de *clusters*. A plataforma disponibiliza um *slider* (barra deslizante) para ajustar o número de *clusters* a ser analisado. O utilizador pode escolher qualquer valor dentro do intervalo predefinido.

A plataforma permite usar uma amostragem dos dados, o que acelera a analise o que é útil em *datasets* grandes e pesados. A plataforma apresenta uma análise detalhada das métricas para diferentes valores de k, permitindo ao utilizador visualizar a variação das métricas antes de decidir a abordagem para determinar o número de *clusters*.

5. **Treinar o Modelo:** Após a definição do número de *clusters*, o utilizador deve clicar no botão "Treinar Modelo Inicial". A plataforma treina o modelo de clustering com o número de *clusters* selecionado e exibe as métricas de desempenho correspondentes. Essas métricas incluem o S*ilhouette Score, Davies-Bouldin Index* e *Calinski-Harabasz Score.*

Após o treino inicial, a plataforma exibe uma caixa para escolher a próxima ação. O utilizador pode optar por re-treinar o modelo ou finalizar a análise.

6. **Re-Treinar o modelo:** Após o treino inicial, o utilizador pode optar por re-treinar o modelo com um novo número de *clusters*. O utilizador pode escolher novamente entre as abordagens automática ou manual para determinar o número de *clusters*. Após definir o novo número de *clusters*, o utilizador deve clicar no botão "Treinar Novamente" para gerar as novas métricas correspondentes.

Após o re-treino, a plataforma exibe novamente as métricas do modelo treinado, permitindo ao utilizador comparar o desempenho com os resultados do treino inicial. O utilizador pode então clicar em "Seguir para Relatório" para prosseguir para a última etapa.

Nota: Caso o utilizador opte por determinar automaticamente o valor de K em ambos os treinos, o resultado será semelhante, uma vez que o melhor valor de K já terá sido definido no primeiro treino. No entanto, recomenda-se que, caso o utilizador escolha a opção automática no primeiro treino, no segundo treino selecione manualmente o valor de K.

2. **Resumo Final do Modelo:** Após o utilizador clicar no botão "Seguir para Relatório", a plataforma gera um relatório final, que inclui:

    - **Modelo Selecionado:** O modelo de *clustering* utilizado (K-*Means* ou *Clustering* Hierárquico).

- **Número de Componentes PCA:** Mostra o número de componentes PCA selecionados.

- **Métricas do Treino Inicial e Re-Treino:** Exibição das métricas comparativas de ambos os treinos. Caso o utilizador tenha optado por re-treinar o modelo, apenas irão aparecer as métricas do treino inicial.

- **Interpretação das Métricas:** A plataforma fornece uma análise dos resultados das métricas obtidas durante os treinos.

- **Gráfico de Comparação de Métricas:** Um gráfico interativo é exibido para comparar as métricas dos modelos treinados. O utilizador pode selecionar a métrica que deseja visualizar no gráfico, facilitando a comparação entre os modelos do treino inicial e do re-treino. Caso o utilizador tenha optado por re-treinar o modelo, apenas irão aparecer as métricas do treino inicial.

- **Indicação do melhor modelo:** A plataforma indica o modelo com o melhor desempenho, com base na métrica *Silhouette Score.*

- ***Download* do Melhor Modelo Treinado:** Após analisar os resultados, o utilizador pode fazer o *download* do melhor modelo treinado. A plataforma permite:

    - ***Download* do Melhor Modelo**: O modelo treinado é salvo com um nome de arquivo, como por exemplo melhor_modelo_treino_inicial.pkl.

    - ***Download* do Relatório PDF**: O utilizador pode fazer o *download* do relatório em PDF contendo todas as métricas, interpretações e comparações feitas durante o treino, para consulta ou apresentação.

Após o *download*, o utilizador pode clicar em "Concluir" para finalizar o processo de treino de modelos.

4. Resultados e Discussão

4.1. Seleção Tempos de Execução de treino de modelos

O desenvolvimento de modelos de *Machine Learning* envolve várias etapas, desde a preparação dos dados até à seleção do algoritmo e configuração dos hiperparâmetros. A Tabela 21 apresenta os tempos de treino registados (eficiência computacional) para cada modelo, refletindo apenas o tempo computacional, sem contabilizar as interações do utilizador, como a escolha do modelo, a definição de parâmetros e a avaliação dos resultados, que podem aumentar significativamente o tempo total de desenvolvimento.

A otimização de hiperparâmetros constitui um processo fundamental na modelação preditiva, influenciando tanto o desempenho do modelo quanto a carga computacional associada ao seu treino. O impacto do *Grid Search* (GS) na eficiência computacional variou consideravelmente entre os modelos, sendo necessário avaliar se os benefícios obtidos justificam o aumento do tempo de processamento. Os tempos de treino registados evidenciam discrepâncias significativas entre os diferentes algoritmos, refletindo a complexidade computacional de cada abordagem e o impacto da otimização dos hiperparâmetros.

Nos modelos de classificação, o K-*Nearest Neighbors (*KNN) apresentou um tempo de treino reduzido, registando 00:00:12 sem GS e 00:00:13 com GS. O impacto da otimização foi, portanto, marginal, sugerindo que a definição manual do parâmetro *k* pode ser suficiente para obter um desempenho satisfatório. O *Random Forest*, por sua vez, demonstrou uma variação mais significativa, passando de 00:00:14 para 00:00:21 com a ativação do GS. Este aumento do tempo de treino deve-se ao ajuste de hiperparâmetros como o número de árvores (*n_estimators*) e a profundidade máxima das mesmas (*max_depth*), que podem influenciar diretamente a complexidade do modelo.

No caso do *Support Vector Classification* (SVC), verificou-se um aumento expressivo do tempo de treino, de 00:04:07 para 00:11:10 com a utilização do GS. Este comportamento evidencia a elevada dependência deste modelo relativamente à escolha dos hiperparâmetros, sendo que a procura exaustiva por combinações ótimas pode implicar um custo computacional significativo. Assim, a otimização de hiperparâmetros pode não ser a abordagem mais eficiente quando se pretende reduzir o tempo total de desenvolvimento.

Nos modelos de regressão, a Regressão Linear Simples (RLS) apresentou um tempo de treino reduzido (00:00:42), em conformidade com a sua menor complexidade computacional e a ausência de hiperparâmetros passíveis de otimização via GS. No entanto, o *Support Vector Regression* (SVR) revelou um comportamento inverso ao do SVC, registando uma redução do tempo de treino de 00:04:18 para 00:03:01 com a ativação do GS. Este resultado sugere que a procura automática conseguiu identificar rapidamente uma configuração eficiente de hiperparâmetros, minimizando o tempo necessário para atingir um desempenho otimizado. Relativamente aos modelos de *clustering*, o K-*Means* registou um tempo de treino de 00:00:38, enquanto o *Clustering* Hierárquico apresentou um tempo significativamente superior (00:03:03). Dado que estes algoritmos não possuem hiperparâmetros ajustáveis pelo método tradicional de GS, a sua eficiência computacional depende predominantemente da quantidade de dados e do número de *clusters* definidos. Métodos auxiliares, como o critério do cotovelo (*Elbow Method*) ou o coeficiente de silhueta, podem ser necessários para determinar empiricamente o número ótimo de *clusters*, o que poderá impactar o tempo total necessário para a configuração do modelo.

A Figura 12 apresenta a variação do tempo de treino entre modelos com e sem GS. Observa-se que no *Support Vector Classification* (SVC), o tempo de treino aumentou significativamente com a ativação do GS, enquanto no *Support Vector Regression* (SVR) ocorreu uma redução do tempo de treino. No caso do *Random Forest*, a diferença foi marginalmente negativa, indicando uma ligeira melhoria no tempo de treino com a otimização automática. Já no KNN, o impacto foi praticamente insignificante. Estes resultados reforçam que o efeito do GS no tempo de treino é altamente dependente da complexidade do modelo e da forma como os hiperparâmetros influenciam o seu desempenho computacional.

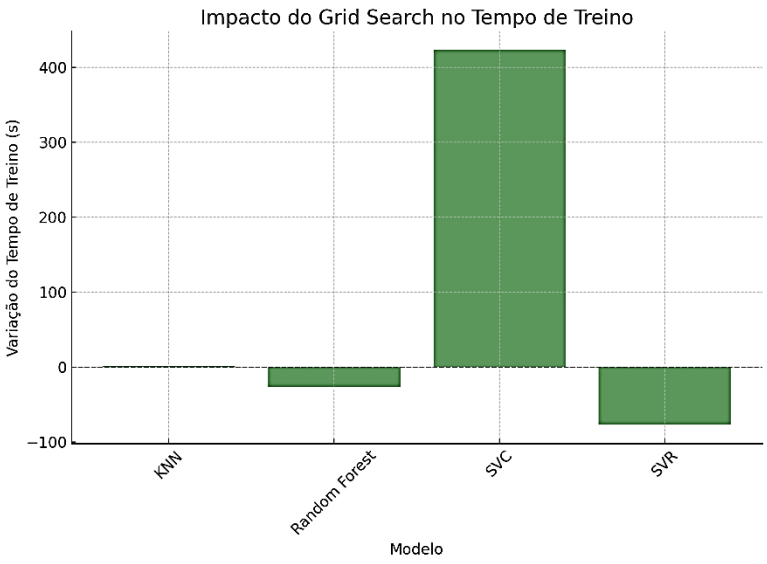


Figura 12. Impacto do *Grid Search* no tempo de treino

A influência do *Grid Search* nos tempos de treino revelou-se altamente variável. No caso do SVR, a otimização de

hiperparâmetros contribuiu para uma redução do tempo de processamento, sugerindo que a procura exaustiva permitiu identificar rapidamente combinações eficientes de hiperparâmetros. Em contrapartida, no SVC, verificou-se um aumento significativo no tempo de treino, indicando que a aplicação do GS pode ser computacionalmente dispendiosa quando aplicada a modelos altamente sensíveis aos hiperparâmetros. Para modelos como *Random Forest* e KNN, o impacto do GS foi moderado, mas ainda assim relevante para avaliar se a otimização automática se justifica face à sua ausência. Assim, a decisão de ativar o GS deve ser ponderada em função da complexidade do modelo e da relação entre o tempo investido na otimização e os benefícios obtidos em termos de desempenho preditivo.

Além disso, a eficiência no desenvolvimento de modelos não depende exclusivamente do tempo computacional, mas também da experiência do utilizador e do suporte de ferramentas avançadas, tais como AutoML, sugestões inteligentes e visualizações interativas. Estas abordagens podem reduzir significativamente o tempo necessário para ajustar e implementar um modelo adequado, tornando o processo de otimização mais eficiente e escalável.

Tabela 21. Tempos de treino dos modelos

Tipo de Modelo	**Modelo**	***Grid Search* Ativado?**	**Primeiro Treino**	**Seleção de *Features***	**Segundo Treino**	**Tempo de treino**
Classificação	K-*Nearest Neighbors* (KNN)	Não	00:00:02	00:00:05	00:00:05	**00:00:12**
		Sim	00:00:03	00:00:05	00:00:05	**00:00:13**
	Random Forest	Sim	00:00:05	00:00:09	00:00:07	**00:00:21**
		Não	00:00:26	00:00:07	00:00:14	**00:00:48**
	Support Vector Classification (SVC)	Não	00:03:22	00:00:07	00:00:38	**00:04:07**
		Sim	00:07:21	00:00:05	00:03:43	**00:11:10**
Regressão	Regressão por Vetores de Suporte (SVR)	Não	00:03:09	00:00:12	00:00:57	**00:04:18**
		Sim	00:02:05	00:00:17	00:00:39	**00:03:01**
	Regressão Linear Simples (RLS)	-	00:00:02	00:00:21	00:00:19	**00:00:42**
Clustering	*KMeans*	-	00:00:19	-	00:00:19	**00:00:38**
	Clustering Hierárquico	-	00:01:39	-	00:01:24	**00:03:03**

4.2. Resultados do treino dos modelos

O treino dos modelos foi realizado na plataforma desenvolvida e posteriormente testado em ambiente *Python*, com o objetivo de comparar os resultados e validar a correspondência entre os desempenhos da aplicação e do código executado num ambiente convencional de *Python*.

A execução dos modelos em Python permitiu uma análise rigorosa das métricas de desempenho, garantindo que os resultados gerados pela aplicação estavam alinhados com aqueles obtidos diretamente pelo código. Foram analisadas as métricas de dois modelos distintos, assegurando consistência e fiabilidade na implementação da plataforma.

Esta abordagem confirmou que a aplicação reproduz corretamente os cálculos e as métricas esperadas, garantindo integridade nos resultados e coerência no desempenho dos modelos, independentemente do ambiente de execução.

4.2.1. Modelo de Classificação – K-*Nearest Neighbors* (KNN)

Os resultados obtidos na plataforma e no ambiente Python apresentam métricas idênticas para o modelo KNN. A distribuição das amostras de treino e teste foi mantida de forma consistente em ambos os ambientes, garantindo comparabilidade nos resultados.

Além disso, os hiperparâmetros foram definidos de forma idêntica, utilizando 5 vizinhos e pesos uniformes, reforçando a fiabilidade da implementação.

As métricas *Accuracy*, *Precision*, *Recall* e F1-*Score*, tanto sem como com seleção de *features*, apresentam diferenças marginais, confirmando que o desempenho do modelo se mantém estável em ambas as abordagens.

Nas Figura 13 e Figura 14, é possível verificar os resultados das métricas em ambos os cenários, evidenciando a equivalência dos cálculos e a coerência entre as implementações.

Dessa forma, conclui-se que a plataforma desenvolvida reproduz com precisão os cálculos e métricas obtidos no ambiente Python, validando assim a sua correta implementação. Consequentemente, pode ser utilizada com confiança para experimentação e análise de modelos baseados em KNN.

Análise dos Resultados do modelo KNN

A análise dos resultados obtidos permite comparar o desempenho do modelo KNN com e sem a seleção de *features*. A principal diferença entre ambas as abordagens reside na escolha das variáveis preditivas utilizadas na classificação, o que pode influenciar as métricas de desempenho do modelo.

- **Sem seleção de *features***: O modelo utilizou todas as 23 variáveis disponíveis, resultando numa *accuracy* de 0.7071, *precision* de 0.6986, *recall* de 0.7071 e *F1-score* de 0.7023.
- **Com seleção de *features***: Apenas as 5 variáveis mais relevantes foram consideradas (Classificação do Imóvel, Distância ao Centro (Km), Preço, Distância ao Metro (Km), Lotação). Este processo levou a uma ligeira redução da *accuracy* para 0.6931, bem como pequenas variações nas demais métricas, com um decréscimo marginal do *F1-score* (0.6886).

A variação reduzida nos resultados sugere que a remoção de determinadas variáveis não compromete significativamente o desempenho do modelo. Tal facto indica que algumas das variáveis excluídas podem conter informações redundantes, e que a seleção de variáveis pode contribuir para uma redução da complexidade computacional sem uma perda expressiva de precisão.

Importa ainda destacar que os hiperparâmetros do modelo foram mantidos constantes em ambas as abordagens (*n_neighbors = 5* e *weights = uniform*), assegurando que as diferenças observadas derivam exclusivamente da seleção das variáveis e não de ajustes nos parâmetros do modelo.

Configurações Utilizadas

Tipo de Modelo: Classificação

Modelo Selecionado: KNeighborsClassifier

Informações dos Conjuntos de Dados

- Amostras de Treino: 25433 (70.0% do total)
- Amostras de Teste: 10900 (30.0% do total)
- Features Originais: 23
- Features Após Seleção: 5

Features Selecionadas

```
[
  0 : "Classificação do Imóvel"
  1 : "Distância do Centro (km)"
  2 : "Preço"
  3 : "Distância do metro (km)"
  4 : "Lotação"
]
```

Comparação de Métricas

	Modelo	Accuracy	Precision	Recall	F1-Score	Best Parameters
0	Sem Seleção de Features	0.7071	0.6986	0.7071	0.7023	{'n_neighbors': 5, 'weights': 'uniform'}
1	Com Seleção de Features	0.6931	0.6850	0.6931	0.6886	{'n_neighbors': 5, 'weights': 'uniform'}

Figura 13. Resultados do KNN na plataforma

```
=== Informações do Conjunto de Dados ===
Amostras de Treino: 25433 (70.00%)
Amostras de Teste: 10900 (30.00%)

Features Selecionadas:
['Classificação do Imóvel', 'Distância do Centro (km)', 'Preço', 'Distância do metro (km)', 'Lotação']

=== Comparação de Métricas ===
                 Modelo  Accuracy  Precision  Recall  F1-Score                         Best Parameters
Sem Seleção de Features    0.7071     0.6986  0.7071    0.7023 {'n_neighbors': 5, 'weights': 'uniform'}
Com Seleção de Features    0.6931     0.6850  0.6931    0.6886 {'n_neighbors': 5, 'weights': 'uniform'}
```

Figura 14. Resultados do KNN em ambiente *python*

4.2.2. Modelo de Regressão – Regressão Linear Simples (RLS)

Os resultados obtidos na plataforma e no ambiente *Python* apresentam métricas idênticas para o modelo de Regressão Linear Simples (RLS). A distribuição das amostras de treino e teste foi mantida de forma consistente, assegurando a comparabilidade dos resultados. Os parâmetros do modelo e as *features* selecionadas foram aplicados da mesma forma nos dois cenários, garantindo que as comparações sejam fiáveis.

As métricas R^2, MAE e MSE, tanto sem como com seleção de *features*, apresentam valores idênticos entre os dois ambientes. Especificamente, a ausência de seleção de *features* resultou num R^2 Score de 0.5535, enquanto a seleção de *features* reduziu esse valor para 0.2719, aumentando os erros MAE e MSE. Isto indica que a remoção de variáveis pode ter impactado negativamente a capacidade preditiva do modelo, reforçando a necessidade de uma seleção criteriosa de *features* para melhorar o desempenho da regressão.

Nas Figura 15 e Figura 16, é possível verificar os resultados das métricas em ambos os cenários, confirmando a equivalência dos cálculos e a coerência entre as implementações.

Assim, conclui-se que a plataforma desenvolvida reproduz corretamente os cálculos e métricas esperados, validando a sua implementação e permitindo a sua utilização para experimentação e análise de modelos de regressão linear.

Análise dos Resultados do modelo RLS

A análise dos resultados permite avaliar o impacto da seleção de *features* no desempenho do modelo de regressão linear. Como esperado, a escolha das variáveis preditivas influência diretamente as métricas de desempenho.

- **Sem seleção de *features***: O modelo utilizou todas as 23 variáveis disponíveis, resultando num coeficiente de determinação (R^2) de 0.5535, um erro absoluto médio (MAE) de 57.2066 e um erro quadrático médio (MSE) de 5618.9704.
- **Com seleção de *features***: Apenas 5 variáveis foram mantidas (Distância do Centro (Km), Casa/Apartamento inteiro, Distância ao metro (Km), Cidade_paris, Cidade_amsterdam), reduzindo o coeficiente R^2 para 0.2719 e aumentando os erros MAE (73.7297) e MSE (9163.3163).

A expressiva redução do coeficiente R^2 e o aumento dos erros indicam que a seleção de *features* impactou negativamente a capacidade preditiva do modelo. Isso sugere que algumas das *features* eliminadas continham informações relevantes para a previsão da variável alvo. A degradação do desempenho reforça a necessidade de uma seleção criteriosa de *features*, garantindo que a remoção de atributos não comprometa significativamente a precisão da regressão.

Além disso, como os parâmetros do modelo foram mantidos constantes nos dois cenários (sem ajuste de hiperparâmetros), pode-se concluir que a diferença nos resultados decorre exclusivamente da seleção de *features* e não de modificações estruturais no modelo.

Figura 15. Resultados do RLS na plataforma

```
=== Informações do Conjunto de Dados ===
Amostras de Treino: 25433 (70.00%)
Amostras de Teste: 10900 (30.00%)

Features Selecionadas:
['Distância do Centro (km)', 'Casa/Apartamento inteiro?', 'Distância do metro (km)', 'Cidade_paris',
'Cidade_amsterdam']

=== Comparação de Métricas ===
                 Modelo  R² Score     MAE       MSE Best Parameters
Sem Seleção de Features    0.5535 57.2066 5618.9704             N/A
Com Seleção de Features    0.2719 73.7297 9163.3163             N/A
```

Figura 16. Resultados do RLS em ambiente *python*

4.3. Considerações finais dos resultados

Os resultados apresentados confirmam que a plataforma desenvolvida reproduz corretamente os cálculos esperados e gera métricas equivalentes às obtidas num ambiente Python convencional, assegurando a sua correta implementação.

Observou-se que modelos mais simples, como KNN e RLS, treinam rapidamente e apresentam baixa dependência da otimização de hiperparâmetros. O KNN demonstrou um impacto marginal com a ativação do GS sugerindo que a definição manual do número de vizinhos (*k*) pode ser suficiente para um bom desempenho. Já a RLS, por não exigir hiperparâmetros otimizáveis, manteve um tempo de treino reduzido e estável.

Por outro lado, modelos mais complexos, como SVC e SVR, mostraram elevada sensibilidade à otimização de hiperparâmetros, refletindo-se em tempos de treino significativamente distintos. O SVC apresentou um aumento expressivo do tempo de treino com GS, enquanto no SVR a otimização permitiu uma redução do tempo de processamento, sugerindo que o GS conseguiu rapidamente encontrar uma configuração eficiente. O *Random Forest* registou uma variação moderada, mostrando que a influência do GS depende da complexidade do modelo e do número de parâmetros a ajustar.

A análise dos resultados dos modelos individuais reforça a fiabilidade da plataforma:

- Para o KNN, a variação nas métricas de desempenho com e sem seleção de *features* foi mínima, indicando que a remoção de algumas *features* não compromete significativamente o desempenho.

- Para a RLS, a remoção de *features* resultou numa queda no coeficiente de determinação (R^2) e num aumento considerável dos erros MAE e MSE, sugerindo que algumas features excluídas continham informações relevantes.

- Para o SVC e o SVR, a dependência de hiperparâmetros e os tempos de treino revelaram-se fatores críticos, sendo necessário um equilíbrio entre otimização e eficiência computacional.

Dessa forma, conclui-se que a plataforma assegura a integridade dos cálculos e a consistência dos resultados obtidos, permitindo a experimentação e análise de diferentes modelos de aprendizagem automática. Além disso, evidencia-se que a decisão de aplicar otimização de hiperparâmetros deve ser criteriosa, considerando a complexidade do modelo e o impacto no tempo de treino face aos benefícios em desempenho preditivo.

5. Conclusão e Trabalhos Futuros

O presente trabalho teve como objetivo o desenvolvimento da MLCASE, uma plataforma automatizada para a construção de modelos de *Machine Learning*, concebida para simplificar e acelerar as etapas envolvidas na criação de modelos preditivos. A plataforma permite que cientistas de dados realizem tarefas essenciais do fluxo de trabalho, incluindo o pré-processamento de dados, a seleção e avaliação de modelos e a otimização de hiperparâmetros, sem necessidade de programação manual intensiva.

Os testes realizados permitiram validar a implementação da plataforma, confirmando que esta consegue reproduzir cálculos compatíveis com um ambiente *Python* convencional**,** garantindo coerência nos resultados obtidos**.** A análise dos tempos de execução demonstrou que a MLCASE apresenta um desempenho satisfatório, permitindo uma utilização eficiente e fluída. A comparação com os resultados obtidos diretamente em *Python* indicou uma boa correspondência dos cálculos, embora haja limitações que devem ser abordadas para tornar a plataforma mais robusta e flexível a diferentes cenários de uso.

Embora a MLCASE apresente vantagens significativas, como a automatização do fluxo de trabalho de *Machine Learning*, uma interface intuitiva e a integração de técnicas de seleção de variáveis e otimização de hiperparâmetros, existem aspetos que podem e devem ser melhorados.

Uma das principais oportunidades de melhoria passa pelo aumento do número de modelos disponíveis, incluindo algoritmos mais avançados, como redes neuronais artificiais, bem como métodos adicionais para clustering e associação de dados. Esta expansão permitirá uma maior flexibilidade na escolha de modelos e melhor adaptabilidade a diferentes tipos de problemas.

Outro aspeto relevante para melhoria diz respeito ao pré-processamento de dados, que poderá ser aprimorado através da integração de técnicas mais completas para tratamento de valores ausentes, deteção de *outliers* e codificação de variáveis categóricas. Além disso, a inclusão de métodos automatizados para redução de dimensionalidade poderá permitir uma otimização mais eficaz dos dados de entrada, melhorando a eficiência computacional dos modelos. A implementação de mecanismos automáticos para a adaptação e organização dos dados poderá contribuir para um pré-processamento mais eficiente, reduzindo a necessidade de intervenção manual e assegurando que os modelos recebam dados otimizados para análise.

A otimização do desempenho da plataforma e dos tempos de treino constitui outra área de melhoria. A adoção de técnicas de paralelização e computação distribuída poderá acelerar significativamente a execução dos modelos, tornando o processo mais eficiente, sobretudo para *datasets* de grande dimensão. Além disso, a gestão de memória e a eficiência do código poderão ser otimizadas para melhorar a escalabilidade da plataforma.

Além destas melhorias técnicas, será relevante aprofundar os estudos sobre tempos de execução, nomeadamente analisando o impacto do tamanho do *dataset* na performance dos modelos e comparando diferentes configurações de *hardware* para identificar as condições mais eficientes para a execução dos algoritmos. Será também pertinente explorar métodos alternativos para a otimização de hiperparâmetros, como *Bayesian Optimization* de forma a tornar o processo mais rápido e eficaz.

Outro aspeto essencial para o desenvolvimento da MLCASE será a recolha de *feedback* dos utilizadores, permitindo compreender melhor as suas necessidades e identificar eventuais dificuldades na utilização da plataforma. Para isso, será fundamental realizar testes com cientistas de dados e profissionais da área, avaliando a usabilidade da interface e a experiência do utilizador. A implementação de um sistema de suporte e documentação mais abrangente contribuirá para uma melhor adoção da plataforma, facilitando a sua utilização por diferentes perfis de utilizadores. Além disso, a inclusão de um módulo de sugestões inteligentes, baseado em inteligência artificial, poderá orientar o utilizador na seleção do modelo mais adequado e na configuração dos parâmetros, tornando a experiência mais intuitiva e acessível.

Em síntese, a MLCASE representa um primeiro passo na automatização do desenvolvimento de modelos de *Machine Learning*, tornando o processo mais acessível e eficiente**.** Os resultados obtidos sugerem que a plataforma tem potencial, mas ainda apresenta limitações que devem ser abordadas para aumentar a sua aplicabilidade em cenários mais exigentes. A expansão das funcionalidades, a otimização do desempenho computacional e a integração com serviços em nuvem são direções futuras que poderão tornar a MLCASE mais robusta e escalável. Por fim, a recolha contínua de *feedback* e o desenvolvimento iterativo da plataforma serão essenciais para garantir a sua evolução e alinhamento com as necessidades do mercado.

Referências Bibliográficas

A Kenneth Reitz Project. (s.d.). *Requests: HTTP for Humans*. Obtido em 10 de Agosto de 2024, de Requests: https://docs.python-requests.org/

A. Sivakumar, R. (2017). A Survey on Data Preprocessing Techniques for. *International Journal of Pure and Applied Mathematics*.

Alexandropoulos, S.-A., Kotsiantis, S., & Vrahatis, M. N. (2019). Data preprocessing in predictive data mining. *The Knowledge Engineering Review*. Obtido em 22 de Fevereiro de 2025, de https://www.researchgate.net/publication/330255711_Data_preprocessing_in_predictive_data_mining

Altair® RapidMiner®. (2024). Obtido em 5 de Novembro de 2024, de Altair: https://altair.com/altair-rapidminer

AltairEngineering. (2025). *Auto Model*. Obtido em 12 de Fevereiro de 2025, de Docs RapidMiner: https://docs.rapidminer.com/latest/studio/guided/auto-model/

Amazon SageMaker. (2024). Obtido em 19 de Setembro de 2024, de AWS AMazon: https://aws.amazon.com/pt/sagemaker/

Anselmo, F. C. (2017). *REGRAS DE ASSOCIAÇÃO – MARKET BASKET ANALYSIS* . Porto. Obtido de https://repositorio-aberto.up.pt/bitstream/10216/107693/2/218342.pdf

Azaank, F. (20 de Aug de 2020). *Como avaliar seu modelo de regressão*. Obtido em 11 de Janeiro de 2025, de Medium: https://medium.com/turing-talks/como-avaliar-seu-modelo-de-regress%C3%A3o-c2c8d73dab96

Azevedo, A., & Santos, M. F. (2008). KDD, SEMMA and CRISP-DM: a parallel overview. *ISCAP - Sistemas de Informação - Comunicações em eventos científicos*. Obtido de https://recipp.ipp.pt/entities/publication/83e28ca0-f825-400c-a00a-9f9c22a36d26

Balgude, S. D., Gite1, S., Pradhan, B., & Lee, C.-W. (2024). Artificial intelligence and machine learning approaches in cerebral palsy diagnosis, prognosis, and management: a comprehensive review. *PeerJ Computer Science*.

Bansal, U. (2014). ECLAT Algorithm for Frequent Item sets Generation. *International Journal of Computer Systems* .

Bergstra, J., & Bengio, Y. (2012). Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*.

Bhandari, A. (29 de Agosto de 2024). *Feature Scaling: Engineering, Normalization, and Standardization*. Obtido em 14 de Novembro de 2024, de Analytics Vidhya: https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning.* Cambridge: Springer Science+Business Media. Obtido de https://github.com/peteflorence/MachineLearning6.867/blob/master/Bishop/Bishop%20-%20Pattern%20Recognition%20and%20Machine%20Learning.pdf

Brandt, J.-M., Benedek, M., Guerin, J. S., & Fliege, J. (2020). Reliability-as-a-Service for Bearing Risk Assessment Investigated with Advanced Mathematical Models. *Internet of Things*.

Breiman, L. (2001). Random Forests. *Springer Science+Business Media LLC, part of Springer Nature*.

Brownlee, J. (2020). *Data Preparation for Machine Learning: Data Cleaning, Feature Selection, and Data Transforms in Python.* Melbourne: Machine Learning Mastery.

Buskirk, T. D. (2018). An Introduction to Machine Learning Methods for Survey Researchers. *Survey Practice, 11*(1), 1-10. doi:https://doi.org/10.29115/SP-2018-0004.

Buzea, C. G., Eva, L., Agop, M., Ochiuz, L., Vasincu, D., Popa, T. O., . . . Buga, R. (2025). AI-Enabled Radiomics and Clinical Decision Support App for Monitoring Brain Metastases Evolution After Gamma Knife Radiosurgery. *PrePrints*. Obtido de https://www.preprints.org/manuscript/202501.0733/v1

Choudhury, A. (17 de Jul de 2024). *What Are Data Types and Why Are They Important?* Obtido em 25 de Fevereiro de 2025, de Amplitude: https://amplitude.com/blog/data-types

Cios, K. J., Swiniarski, R. W., Pedrycz, W., & Kurgan, L. A. (2007). Unsupervised Learning: Association Rules. Em K. J. Cios, R. W. Swiniarski, W. Pedrycz, & L. A. Kurgan, *Data Mining- A Knowledge Discovery Approach* (p. 289). Boston: Springer New York, NY. doi:https://doi.org/10.1007/978-0-387-36795-8

Concannon, M. (29 de Aug de 2024). *Understanding Machine Learning: A Beginner's Guide*. Obtido em 11 de Dezembro de 2024, de Ntiva: https://www.ntiva.com/blog/what-is-machine-learning

Costa, B. (11 de Dec de 2019). *Uma Introdução ao Algoritmo Apriori*. Obtido em 13 de Junho de 2024, de Medium: https://medium.com/@bernardo.costa/uma-introdu%C3%A7%C3%A3o-ao-algoritmo-apriori-60b11293aa5a

Daza, A., Rueda, N. D., Sánchez, M. S., Espíritu, W. F., & Quiñones, M. E. (2024). Sentiment Analysis on E-Commerce Product Reviews Using Machine Learning

and Deep Learning Algorithms: A Bibliometric Analysis, Systematic Literature Review, Challenges and Future Works. *Data Insights*.

Duarte, R. (30 de Nov de 2023). *Métricas de Avaliação em Modelos de Classificação em Machine Learning*. Obtido de Sigmoidal: https://sigmoidal.ai/metricas-de-avaliacao-em-modelos-de-classificacao-em-machine-learning/

Duarte, R. (1 de Nov de 2023). *Métricas de Avaliação em Modelos de Regressão em Machine Learning*. Obtido de Sigmoidal: https://sigmoidal.ai/metricas-de-avaliacao-em-modelos-de-regressao-em-machine-learning/

Duda, L. J., Woiciechowski, A. L., Scapini, T., & Soccol, C. R. (2024). Advancements in Thermochemical Biomass Conversion: AI Techniques. ResearchGate. *Congresso Brasileiro de Biotecnologia Industrial.* Florianópolis, Santa Catarina/BR.

Enterprise. (2024). *Enterprise Generative AI*. Obtido em 08 de Agosto de 2024, de h2o: https://h2o.ai/platform/enterprise-h2ogpte/

Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining* .

Extraction of the essential elements for urban systems modelling – A word-to-vector approach. (2024). *City and Environment*. Obtido de https://www.sciencedirect.com/science/article/pii/S2590252024000266

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J. T., Blum, M., & Hutter, F. (2015). *Efficient and Robust Automated Machine Learning.* Obtido de NeurIPS Proceedings: https://proceedings.neurips.cc/paper_files/paper/2015/file/11d0e6287202fced83f79975ec59a3a6-Paper.pdf

Filho, M. (1 de Jan de 2023). *O Que É Acurácia Em Machine Learning?* Obtido em 10 de Janeiro de 2025, de mariofilho: https://mariofilho.com/o-que-e-acuracia-em-machine-learning/#:~:text=Acur%C3%A1cia%20%C3%A9%20uma%20m%C3%A9trica%20de%20avalia%C3%A7%C3%A3o%20muito%20popular,de%20previs%C3%B5es%20corretas%20pelo%20n%C3%BAmero%20total%20de%20previs%C3%B5es.

Fontelles, E. (2024). *Como melhor tratar variáveis categóricas para modelos de machine learning — Parte 1*. Obtido em 22 de Fevereiro de 2025, de Datarisk: https://www.datarisk.io/como-melhor-tratar-variaveis-categoricas-para-modelos-de-machine-learning-parte-1/?utm_source=chatgpt.com

Francisco, A. M. (2024). *Aplicação De Modelos De Machine Learning Para Previsão Do Housing Price Em Singapura.* Lisboa: ISCTE - Instituto Universitário de Lisboa.

Obtido de https://repositorio.iscte-iul.pt/bitstream/10071/33305/1/Master_antonio_muinga_francisco.pdf

Freiburg, P. (2022). *auto-sklearn*. Obtido em 2025 de Fevereiro de 2025, de Automl.gitgub: https://automl.github.io/auto-sklearn/master/#manual

Geeks for Geek. (25 de January de 2024). *ML | Overview of Data Cleaning*. Obtido em 17 de Fevereiro de 2025, de geeks for geek: https://www.geeksforgeeks.org/data-cleansing-introduction/?ref=lbp

Geeks for geeks. (15 de Jul de 2024). *K-Nearest Neighbor(KNN) Algorithm*. Obtido em 7 de Janeiro de 2025, de Geeks for geeks: https://www.geeksforgeeks.org/k-nearest-neighbours/

Geeks for Geeks. (27 de February de 2024). *Supervised Machine Learning*. Obtido em 12 de Fevereiro de 2025, de Geeks for geeks: https://www.geeksforgeeks.org/supervised-machine-learning/

GeeksforGeeks. (05 de Nov de 2023). *Davies-Bouldin Index*. Obtido em 1 de Fevereiro de 2025, de Geeks for geeks: https://www.geeksforgeeks.org/davies-bouldin-index/

GeeksforGeeks. (2024). *PyTorch Tutorial - Learn PyTorch with Examples*. Obtido em 12 de Fevereiro de 2025, de Geeks for geeks: https://www.geeksforgeeks.org/pytorch-learn-with-examples/

Géron, A. (2017). *Hands-On Machine Learning with Scikit-Learn and TensorFlow.* Sebastopol: O'Reilly. Obtido de https://www.clc.hcmus.edu.vn/wp-content/uploads/2017/11/Hands_On_Machine_Learning_with_Scikit_Learn_and_TensorFlow.pdf

GitHub. (2025). *fpdf2*. Obtido em 28 de Janeiro de 2025, de Py-pdf: https://py-pdf.github.io/fpdf2/

Google. (2024). *AutoML*. Obtido em 11 de Fevereiro de 2025, de Google Cloud: https://cloud.google.com/automl

Google Cloud. (2024). *Guia do AutoML para iniciantes*. Obtido em 04 de Dezembro de 2024, de Google Cloud: https://cloud.google.com/vertex-ai/docs/beginner/beginners-guide

Gültekin, H. (7 de Sep de 2023). *What is Silhouette Score?* Obtido em 3 de Julho de 2024, de Medium: https://medium.com/@hazallgultekin/what-is-silhouette-score-f428fb39bf9a

Guyon, I., Nikravesh, M., Gunn, S., & Zadeh, L. A. (2008). *Feature Extraction - Foundations and Applications.* The Netherlands: Springer Berlin, Heidelberg. doi:https://doi.org/10.1007/978-3-540-35488-8

Hamza, M. F., & Sjarif, N. N. (2024). A Comprehensive Overview of Heart Sound. *IEEE Xplore*.

Han, J., Kamber, M., & Pei, J. (2012). *Data Mining: Concepts and Techniques.* Waltham: Morgan Kaufmann Publishers.

Haque, M. Z., & Sarker, Y. (2025). Sentiment Classification of Product Reviews in the Bangla Language Using NLP and Machine Learning. *ResearchGate*.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning - Data Mining, Inference, and Prediction.* New York: Springer Science+Business Media. Obtido de https://esl.hohoweiya.xyz/book/The%20Elements%20of%20Statistical%20Learning.pdf

Hofmann, M., & Klinkenberg, R. (2014). *RapidMiner - Data Mining Use Cases and Business Analytics Applications.* Boca Raton: CRC Press. Obtido de https://api.pageplace.de/preview/DT0400.9781482205503_A24097365/preview-9781482205503_A24097365.pdf

Hutter, F., Kotthoff, L., & Vanschoren, J. (2019). *Automated Machine Learning. Methods, Systms, Challenges.* Switzerland: The Springer Series.

Hyperparameters of Random Forest Classifier. (22 de Jan de 2021). Obtido em 29 de Janeiro de 2025, de Geeksforgeeks: https://www.geeksforgeeks.org/hyperparameters-of-random-forest-classifier/

IBM. (2025). *Árvore de regressão*. Obtido em 16 de Fevereiro de 2025, de IBM: https://www.ibm.com/docs/pt-br/cognos-analytics/12.0.0?topic=tests-regression-tree

IBM. (2025). *What is the k-nearest neighbors (KNN) algorithm?* Obtido em 17 de Fevereiro de 2025, de IBM: https://www.ibm.com/think/topics/knn

IBM. (2025). *What is the k-nearest neighbors (KNN) algorithm?* Obtido em 17 de Fevereiro de 2025, de IBM: https://www.ibm.com/think/topics/knn

Jaadi, Z. (23 de Feb de 2024). *Principal Component Analysis (PCA): A Step-by-Step Explanation*. Obtido de Builtin: https://builtin.com/data-science/step-step-explanation-principal-component-analysis

Joblib developers. (2021). *Joblib: running Python functions as pipeline jobs*. Obtido em 18 de Agosto de 2024, de Joblib: https://joblib.readthedocs.io/en/stable/

Junior, E. (29 de Jul de 2023). *Principais métricas de avaliação de modelos em Machine Learning*. Obtido de Medium: https://medium.com/data-hackers/principais-m%C3%A9tricas-de-classifica%C3%A7%C3%A3o-de-modelos-em-machine-learning-94eeb4b40ea9

Kaggle. (4 de Novembro de 2024). *Intro to Machine Learning*. Obtido de Kaggle: https://www.kaggle.com/code/dansbecker/how-models-work

Katsumbe, T. H., Telukdarie, A., Munsamy, M., & Tshukudu, C. (2024). Extraction of the essential elements for urban systems modelling – A word-to-vector approach. *City and Environment*.

Kiwa, F. J., Mnkandla, A. Z., Ndlovu, B. M., Dube, S., & Nyoni, P. (2024). Loan Eligibility System Using Machine Learning and Streamlit Framework. *7th European Conference on Industrial Engineering and Operations.* Germany. Obtido de https://www.researchgate.net/publication/384801399_Loan_Eligibility_System_Using_Machine_Learning

Kopanja, M., Savic, M., & Longo, L. (2025). CORTEX: A COST-SENSITIVE RULE AND TREE EXTRACTION. *arXiv*. Obtido de https://arxiv.org/pdf/2502.03200

Koray, A.-M., Gyimah, E., Metwally, M., Rahnema, H., & Tomomewo, O. (2025). Leveraging machine learning for enhanced reservoir permeability estimation in geothermal hotspots: a case study of the Williston Basin. *Geotherm Energy, 13*. doi:10.1186/s40517-024-00323-4

Lane, M., Maiocco, A., Bhatia, S. K., & Climer, S. (2020). Chapter Four - Eyeing the patterns: Data visualization using doubly-seriated color heatmaps. *Advances in Computers*.

Li, B., Eisenberg, N., Beaton, D., Lee, D. S., Al-Omran, L., Wijeysundera, D. N., . . . Al-Omran, M. (2024). Predicting lack of clinical improvement following varicose vein ablation using machine learning. *Journal Pre-proof*.

Li, H., Tan, Y., Zeng, D., Su, D., & Qiao, S. (2025). Attitude-Predictive Control of Large-Diameter Shield Tunneling: PCA-SVRMachineLearning Algorithm Application in a Case Study of the Zhuhai Xingye Express Tunnel. *Applied Sciences*. Obtido de https://www.mdpi.com/2076-3417/15/4/1880

Lundberg, S. M., & Lee, S.-I. (25 de Nov de 2017). *A Unified Approach to Interpreting Model.* Obtido de Arxiv: https://arxiv.org/pdf/1705.07874

Machado, P. F. (2022). *Conception and Evaluation of Data Augmentation techniques for Tabular Data.* Universidade do Minho,Escola de Engenharia.

Mariscal, G., Marban, O., & Fernandez, C. (2010). A survey of data mining and knowledge discovery process models and methodologies. *The Knowledge Engineering Review*. doi:10.1017/S026988891000003

Mhadhbi, N. (29 de Set de 2024). *Streamlit Python: Tutorial*. Obtido em 29 de Dezembro de 2024, de Data Camp: http://datacamp.com/tutorial/streamlit

Microsoft. (2024). *MLOps model management with Azure Machine Learning*. Obtido em 12 de Fevereiro de 2025, de Learn Microsoft: https://learn.microsoft.com/en-us/azure/machine-learning/concept-model-management-and-deployment?view=azureml-api-2

Microsoft. (2024). *What is Azure Machine Learning?* Obtido em 16 de Setembro de 2024, de Learn Microsoft: https://learn.microsoft.com/en-us/azure/machine-learning/overview-what-is-azure-machine-learning?view=azureml-api-2

Mlakar, U., Jr., I. F., & Fister, I. (30 de Dec de 2024). *NiaAutoARM: Automated generation and evaluation of Association Rule Mining pipelines.* Obtido em 1 de Fevereiro de 2025, de arxiv: https://arxiv.org/pdf/2501.00138

MLJAR. (s.d.). *MLJAR Studio*. Obtido em 1 de Fevereiro de 2025, de Mljar: https://mljar.com/

Modelos de machine learning. (2024). Obtido em 13 de Setembro de 2024, de databricks: https://www.databricks.com/br/glossary/machine-learning-models

Mostafa, R. R., Taha, N. M., & Eissa, F. M. (2024). Artificial intelligence in Medical Parasitology diagnosis and drug discovery: A systematic review. *Parasitologists United Journal.*

Narvekar, M., & Syed, S. F. (2015). An optimized algorithm for association rule mining using FP tree. *International Conference on Advanced Computing Technologies and Applications.* doi:https://doi.org/10.1016/j.procs.2015.03.097

NumPy Developers. (2024). *NumPy documentation*. Obtido em 1 de Dezembro de 2024, de NumPy: https://numpy.org/doc/stable/

O'Reilly. (2025). *Chapter 4. Supervised Learning: Models and Concepts*. Obtido em 25 de Fevereiro de 2025, de Oreilly: https://www.oreilly.com/library/view/machine-learning-and/9781492073048/ch04.html

Obaido, G., Mienye, I. D., Egbelowo, O. F., Emmanuel, I. D., Ogunleye, A., Ogbuokiri, B., . . . Aruleba, K. (2024). Supervised machine learning in drug discovery and development: Algorithms, applications, challenges, and prospects. *Machine Learning with Applications*.

Olson, R. S., Bartley, N., Urbanowicz, R. J., & Moore, J. H. (20 de Mar de 2016). *Evaluation of a Tree-based Pipeline Optimization Tool*. Obtido de arxiv: https://arxiv.org/pdf/1603.06212

Omran, M., Engelbrecht, A., & Salman, A. A. (2007). An overview of clustering methods. *Intelligent Data Analysis* .

Pachyderm. (2023). *Data-driven pipelines*. Obtido em 26 de Setembro de 2024, de Pachyderm: https://www.pachyderm.com/

Pandas. (2024). Obtido em 2 de Dezembro de 2024, de Pandas Documentation: https://pandas.pydata.org/docs/index.html

Plotly. (2024). *Plotly Open Source Graphing Library for Python*. Obtido em 11 de Dezembro de 2024, de Plotly: https://plotly.com/python/

Python Software Foundation. (2025). *datetime — Basic date and time types*. Obtido em 14 de Janeiro de 2025, de Docs Pyhton: https://docs.python.org/3/library/datetime.html

Python Software Foundation. (2025). *decimal — Decimal fixed-point and floating-point arithmetic*. Obtido em 17 de Janeiro de 2025, de Docs Python: https://docs.python.org/3/library/decimal.html

Python Software Foundation. (2025). *fractions — Rational numbers*. Obtido em 11 de Fevereiro de 2025, de Docs Python: https://docs.python.org/3/library/fractions.html

Python Software Foundation. (Jan de 2025). *io — Core tools for working with streams*. Obtido de Docs Python: https://docs.python.org/3/library/io.html

Python Software Foundation. (2025). *os — Miscellaneous operating system interfaces*. Obtido em 7 de Fevereiro de 2025, de Docs Python: https://docs.python.org/3/library/os.html

Python Software Foundation. (2025). *pickle — Python object serialization*. Obtido em 19 de Janeiro de 2025, de Docs Python: https://docs.python.org/3/library/pickle.html

Python Software Foundation. (2025). *re — Regular expression operations*. Obtido de Docs Python: https://docs.python.org/3/library/re.html

Python Software Foundation. (2025). *tempfile — Generate temporary files and directories*. Obtido de Docs Python: https://docs.python.org/3/library/tempfile.html

Python Software Foundation. (2025). *time — Time access and conversions*. Obtido em 27 de Janeiro de 2025, de Docs Pyhton: https://docs.python.org/3/library/time.html

QuinnRadich, V-alje, & Eliotcowley. (18 de Abr de 2024). *O que é um modelo de machine learning?* Obtido em 14 de Dezembro de 2024, de Microsoft Learn: https://learn.microsoft.com/pt-pt/windows/ai/windows-ml/what-is-a-machine-learning-model

Ranjbarzadeh, R., Caputo, A., Tirkolaee, E. B., Ghoushchi, S. J., & Bendechache, M. (2023). Brain tumor segmentation of MRI images: A comprehensive review on the application of artificial intelligence tools. *Computers in Biology and Medicine*.

Raschka, S. (2023). *Apriori: Frequent itemsets via the Apriori algorithm*. Obtido em 1 de Fevereiro de 2025, de Rasbt github: https://rasbt.github.io/mlxtend/user_guide/frequent_patterns/apriori/

Raschka, S. (2023). *Fpgrowth: Frequent itemsets via the FP-growth algorithm*. Obtido em 1 de Fevereiro de 2025, de Rasbt github: https://rasbt.github.io/mlxtend/user_guide/frequent_patterns/fpgrowth/

Raschka, S. (2023). *MLxtend*. Obtido em 1 de Fevereiro de 2025, de rasbt.github: https://rasbt.github.io/mlxtend/

Razzano, G., Brandi, S., Piscitelli, M. S., & Capozzoli, A. (2025). Ruleextraction fromdeepreinforcementlearningcontroller and. *Elsevier*. Obtido de https://www.sciencedirect.com/science/article/pii/S0306261924024309

Ribeiro, M. T., Singh, S., & Guestrin, C. (9 de 08 de 2016). *"Why Should I Trust You?" Explaining the Predictions of Any Classifier*. Obtido de Arxiv: https://arxiv.org/pdf/1602.04938

Rizvi, S. M. (9 de Aug de 2024). *Gini Index & Entropy: 2 Impurity Measures*. Obtido em 23 de Fevereiro de 2025, de Datasciencedojo: https://datasciencedojo.com/blog/gini-index-and-entropy/

Ruan, X., Sun, H., Shou, W., & Wang, J. (2024). The Impact of Climate Change and Urbanization on Compound Flood Risks in Coastal Areas: A Comprehensive Review of Methods. *Applied Sciences*.

S.Olson, R. (2024). *TPOT*. Obtido em 17 de Novembro de 2024, de Epistasislab: https://epistasislab.github.io/tpot/

Salimi, A., Ghobrial, T., & Bonakdari, H. (2024). valuating data preparation techniques on machine learning forecasting of Ice Jam Flood occurrence: A case study on the Chaudière River. *27th IAHR International Symposium on Ice.* Gdansk, Poland.

Salunke, D. (28 de Aug de 2023). *SVC (Support Vector Classifier)*. Obtido em 29 de Julho de 2024, de Linkedin: https://www.linkedin.com/pulse/svc-support-vector-classifier-dishant-salunke/

Scikit-learn developers. (2025). *6. Dataset transformations*. Obtido em 1 de Fevereiro de 2025, de Scikit Learn: https://scikit-learn.org/stable/modules/preprocessing.html#scaling-features-to-a-range

Scikit-learn Developers. (2025). *AgglomerativeClustering*. Obtido em 1 de Fevereiro de 2025, de Scikit-learn: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html

Scikit-learn Developers. (2025). *Decision Trees*. Obtido em 18 de Fevereiro de 2025, de Scikit-Learn: https://scikit-learn.org/stable/modules/tree.html

Scikit-learn Developers. (2025). *KMeans*. Obtido em 15 de Janeiro de 2025, de Scikit-learn: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

Scikit-learn Developers. (2025). *KNeighborsClassifier*. Obtido em 7 de Janeiro de 2025, de Scikit-learn: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

Scikit-learn Developers. (2025). *LinearRegression*. Obtido em 1 de Fevereiro de 2025, de Scikit-learn: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

Scikit-learn Developers. (2025). *Scikit-learn Machine Learnign in Python*. Obtido em 1 de Fevereiro de 2025, de Scikit-learn: https://scikit-learn.org/stable/getting_started.html

Scikit-learn Developers. (2025). *SVC*. Obtido em 17 de Janeiro de 2025, de Scikit-learn: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

Scikit-learn Developers. (2025). *SVR*. Obtido em 15 de Fevereiro de 2025, de Scikit-learn: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html

Shimaoka, A. M., Ferreira, R. C., & Goldman, A. (2024). The evolution of CRISP-DM for Data Science: Methods, Processes and Frameworks. *SBC Reviews on Computer Science*. doi:10.5753/reviews.2024.3757

Siddiqui, A., & Sorrot, R. (2024). *Multiple Disease Prediction.* Solan, India.

Silva, D. H., Souza, L., Ribeiro, C., Brasileiro, S., Nardo, J., Pereira, A., & Andrade, A. (2024). A Web Application for exploratory data analysis and classification of Parkinson's Disease patients using machine learning models on different datasets. *Software Impacts*.

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical Bayesian Optimization of Machine. Em *Advances in Neural Information Processing Systems 25.* Obtido de https://papers.nips.cc/paper_files/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf

Snowflake. (2024). *A faster way to build and share data apps*. Obtido em 20 de Janeiro de 2025, de Streamlit: https://streamlit.io/

Snowflake Inc. (2025). *Install Streamlit*. Obtido em 19 de Janeiro de 2025, de Docs Streamlit: https://docs.streamlit.io/get-started/installation

Sousa, B. (2025). *MLCase - Plataforma de Machine Learning*. Obtido em 1 de Março de 2025, de mlcase-plataform: https://mlcase-plataform.streamlit.app/

Streamlit. (2025). *Streamlit documentation*. Obtido em 27 de Fevereiro de 2025, de Streamlit: https://docs.streamlit.io/

Sun, J., Du, W., & Shi, N. (2018). A Survey of kNN Algorithm.

Sundararaj, K. (2024). MobileNet and Streamlit Applications for Classification and Prediction of MRI-Based Types of Brain Tumor. *International Journal of Advanced Trends in Engineering and Management (IJATEM)*. Obtido de https://ijatem.com/wp-content/uploads/2024/08/IJATEM24MAY002.pdf

Supervised and Unsupervised learning. (04 de December de 2023). Obtido de geeks fo rgeeks: https://www.geeksforgeeks.org/supervised-unsupervised-learning/

Supervised and Unsupervised learning. (04 de December de 2023). Obtido em 3 de Janeiro de 2025, de geeks for geeks: https://www.geeksforgeeks.org/supervised-unsupervised-learning/

Tech, D. (2024). *O que é TensorFlow? Para que serve?* Obtido em 12 de Fevereiro de 2025, de Didatica Tech: https://didatica.tech/o-que-e-tensorflow-para-que-serve/#:~:text=TensorFlow%20%C3%A9%20uma%20biblioteca%20de%20c%C3%B3digo%20aberto%20criada,principais%20ferramentas%20para%20machine%20learning%20e%20deep%20learning.

The AI-Native WEKA Data Platform. (2024). Obtido de Weka: https://www.weka.io/product/ai-native-data-platform/

The Matplotlib development team. (2024). *Matplotlib 3.10.0 documentation*. Obtido em 17 de Dezembro de 2024, de matplotlib: https://matplotlib.org/stable/index.html

The SciPy community. (2025). *SciPy documentation*. Obtido em 28 de Janeiro de 2025, de Docs SciPy: https://docs.scipy.org/doc/scipy/

Theodoridis, S., & Koutroumbas, K. (2009). *Chapter 15 - Clustering Algorithms IV*. Greece: Elsevier.

Tiwari, P., & Park, K.-I. (2024). Seed Biotechnologies in Practicing Sustainable Agriculture: Insights and Achievements in the Decade 2014–2024. *Applied Sciences*.

Unsupervised Learning. (04 de December de 2023). Obtido em 16 de Julho de 2024, de geeks for geeks: https://www.geeksforgeeks.org/ml-types-learning-part-2/?ref=next_article

Ustebay, S., Turgut, Z., & Aydin, M. A. (2018). Intrusion Detection System with Recursive Feature. *International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism.* Ankara, Turkey.

Wang, J., Wang, Y., Li, G., & Qi, Z. (2024). Integration of Remote Sensing and Machine Learning for Precision Agriculture: A Comprehensive Perspective on Applications. *MDPI Agronomy*.

Waskom, M. (2024). *seaborn: statistical data visualization*. Obtido em 28 de Dezembro de 2024, de seaborn: https://seaborn.pydata.org/

Wei, H. (2 de Jan de 2020). *How to measure clustering performances when there are no ground truth?* Obtido de Medium: https://medium.com/@haataa/how-to-measure-clustering-performances-when-there-are-no-ground-truth-db027e9a871c

WekaIO. (2024). *The AI-Native WEKA Data Platform*. Obtido em 30 de Novembro de 2024, de Weka: https://www.weka.io/product/ai-native-data-platform/

Xin He, K. Z. (16 de 04 de 2021). *AutoML: A Survey of the State-of-the-Art.* Obtido em 22 de Novembro de 2024, de Arxiv: https://arxiv.org/pdf/1908.00709.pdf

Youness, G., Phan, N. U., & Boulakia, B. C. (2023). BootBOGS: Hands-on optimizing Grid Search in. *ICCSA 2023: 20th ACS/IEEE International Conference on Computer Systems and Applications, ACS/IEEE International Conference on Computer Systems and Applications*.

Zermane, H., Zermane, A., & Tohir, M. Z. (2024). MACHINE LEARNING TECHNIQUES FOR FATAL ACCIDENT PREDICTION. *ACC Jornal*.

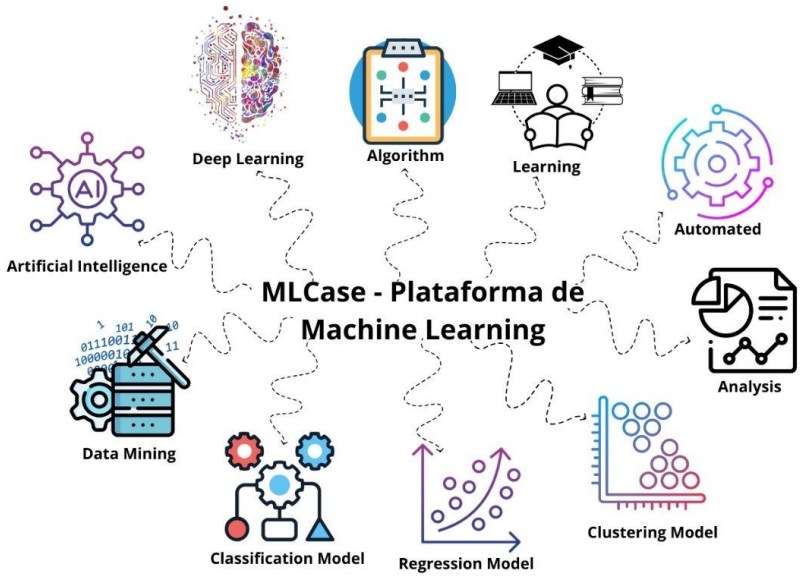
Zhang, D., Theeramunkong, T., & Khamsemanan, N. (2024). Predicting China's Marriage Rate using Machine Learning Models and Cross-Validation Evaluation. *nternational Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC).* doi:10.1109/CyberC62439.2024.00045

Zuhdi, D. A., Sulistyanto, A., & Harahap, H. S. (2024). Dynamic of gender-based violence in Scopus Indexed Publication 2014-2024: A bibliometric analysis . *Journal of Applied Retail Analytics*.

Anexo A - Guia de Utilizador

MLCASE - PLATAFORMA DE MACHINE LEARNING

GUIA DE UTILIZAÇÃO



https://mlcase-plataform.streamlit.app/

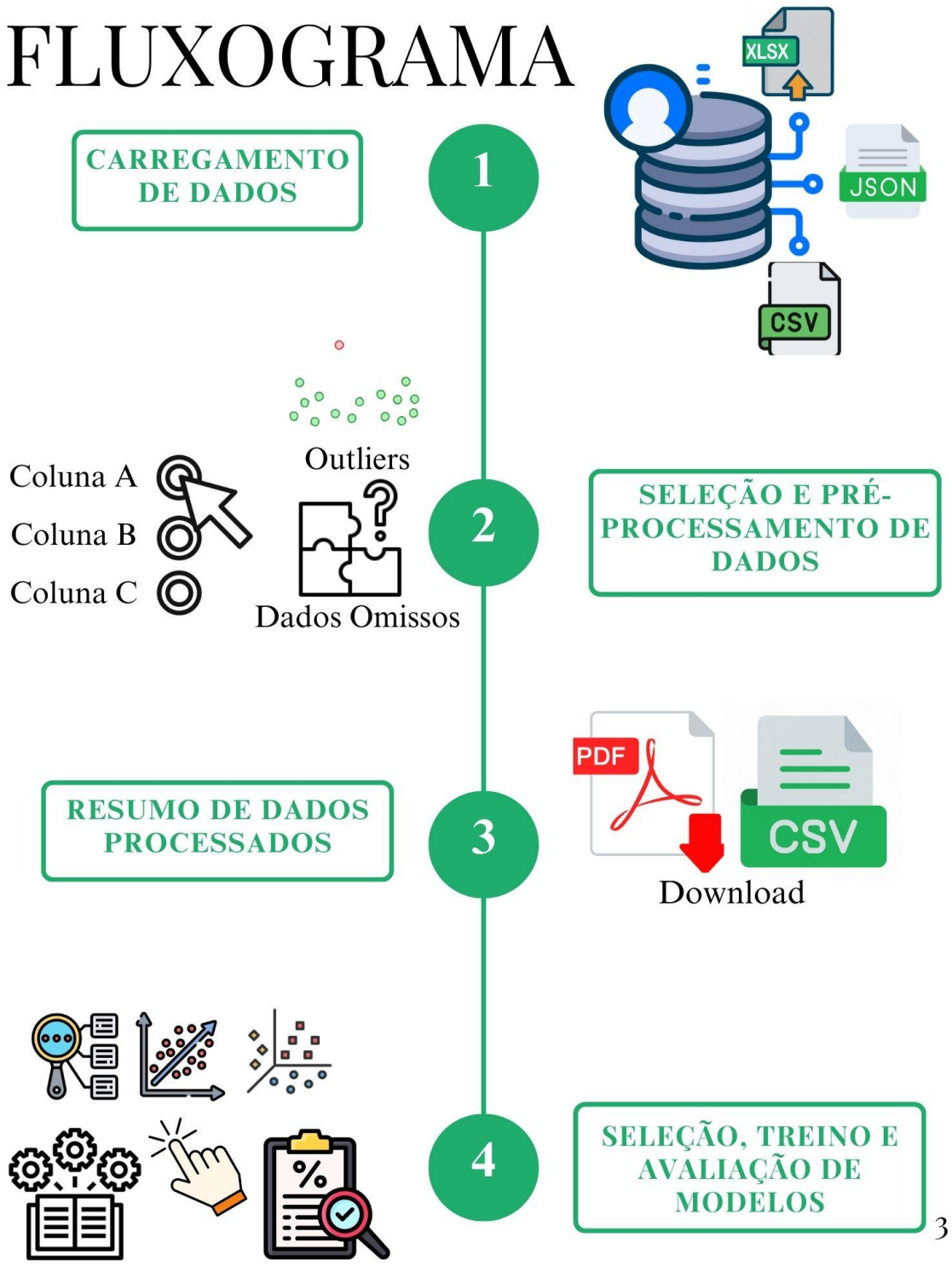
Bruna Sousa

OBJETIVO

A plataforma MLCase - Plataforma de Machine Learning automatiza diversas etapas do processo de ML, tornando o trabalho dos cientistas de dados mais eficiente e acelerando a criação de modelos. A construção de modelos de ML normalmente envolve várias fases, como o pré-processamento dos dados, a construção e avaliação dos modelos. A automatização dessas fases pode poupar tempo e esforço, permitindo que os cientistas de dados se dediquem mais à análise e à interpretação dos resultados.

Com esta plataforma, procura-se reduzir a carga de trabalho repetitiva e agilizar o desenvolvimento de modelos, permitindo que os cientistas de dados se concentrem em tarefas mais complexas, como interpretar os resultados e gerar *insights* significativos.

2


FLUXOGRAMA
CARREGAMENTO DE DADOS
1
XLSX
JSON
CSV
Outliers
Coluna A
Coluna B
Coluna C
Dados Omissos
2
SELEÇÃO E PRÉ-PROCESSAMENTO DE DADOS
RESUMO DE DADOS PROCESSADOS
3
PDF
CSV
Download
4
SELEÇÃO, TREINO E AVALIAÇÃO DE MODELOS

3

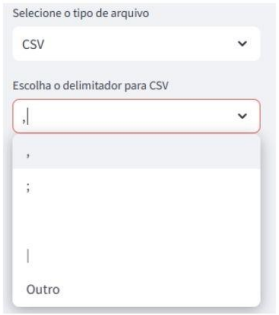
01 CARREGAMENTO DE DADOS

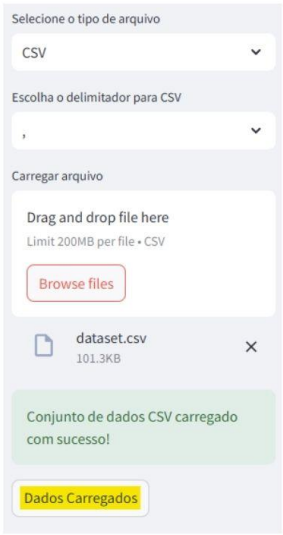
A plataforma permite o carregamento de ficheiros em 3 formatos diferentes:

- CSV
- Excel
- Json

Para o formato de CSV, é necessário escolher o delimitador das colunas do ficheiro. Existem as opções de vírgula (,), ponto e vírgula (;), tabulação ( ), barra vertical (|) ou outro, onde pode escolher outro delimitador que não esta nas opções.

Após o upload do ficheiro é necessário carregar no botão de Dados Carregados para seguir para o passo seguinte.





4

02 SELEÇÃO E PRÉ-PROCESSAMENTO DE DADOS

Esta etapa está dividida em 3 subetapas sendo ela:

- Seleção, Identificação e Discretização de variáveis
- Tratamento de valores ausentes
- Deteção de Outliers

5

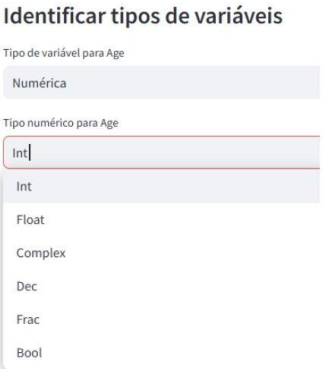
02.1 SELEÇÃO, IDENTIFICAÇÃO E DISCRETIZAÇÃO DE VARIÁVEIS

Seleção de Colunas

A plataforma exibe uma visualização preliminar do conjunto de dados carregado. Abaixo da tabela de visualização, são listadas todas as colunas selecionadas por padrão. Caso deseje remover alguma coluna, basta clicar no ícone de remoção (representado por um "X"). Se, por algum motivo, remover uma coluna acidentalmente, pode clicar na seta para baixo para aceder à lista de colunas removidas. Ao clicar sobre a coluna desejada, ela será adicionada novamente ao conjunto de dados.

Identificar Tipo de variáveis

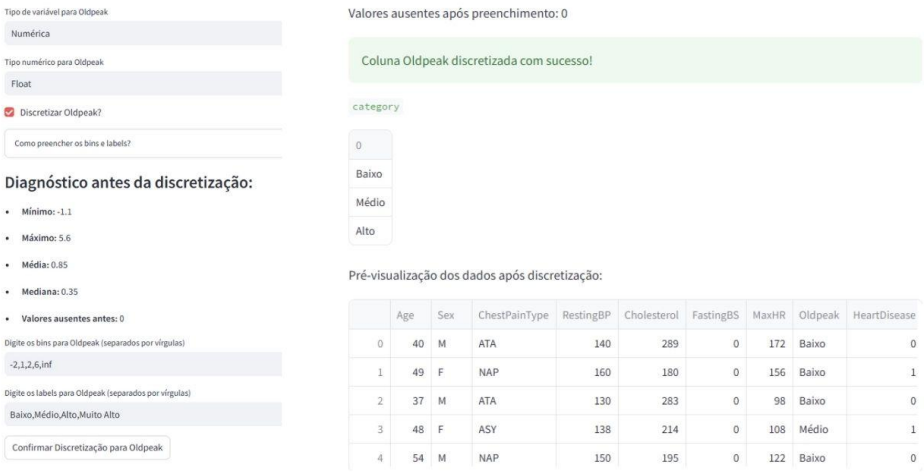
O código detecta automaticamente o tipo de dados de cada coluna e preenche o tipo de variável correspondente. No entanto, você pode modificar o tipo de variável sugerido, caso necessário. As opções de tipos de variáveis disponíveis são: Categórica, Data ou Numérica. Se a coluna for numérica, será necessário escolher o tipo de dado numérico, como: Int, Float, Complex, Dec, Frac ou Bool.



02.1 SELEÇÃO, IDENTIFICAÇÃO E DISCRETIZAÇÃO DE VARIÁVEIS

Discretização Variáveis

Se você optar por discretizar uma variável numérica, a plataforma converte os valores contínuos em categorias discretas, utilizando intervalos . Antes de realizar a discretização, a plataforma exibe um diagnóstico da coluna selecionada, fornecendo informações importantes sobre os dados. Essas informações são úteis para que compreenda a distribuição dos dados antes de aplicar a discretização. O código sugere automaticamente intervalos e rótulos com base na análise dos dados. No entanto, pode modificar essas sugestões conforme necessário para adaptar melhor a discretização ao contexto.
Existe uma explicação na plataforma de como preencher os *bins* e *labels.*



7

02.2 TRATAMENTO DE VALORES AUSENTES

A plataforma identifica automaticamente as colunas com valores ausentes e exibe uma tabela com o número de valores ausentes e os nomes das colunas. Se não houver valores ausentes nos dados, será exibida uma mensagem informando que não há valores ausentes.

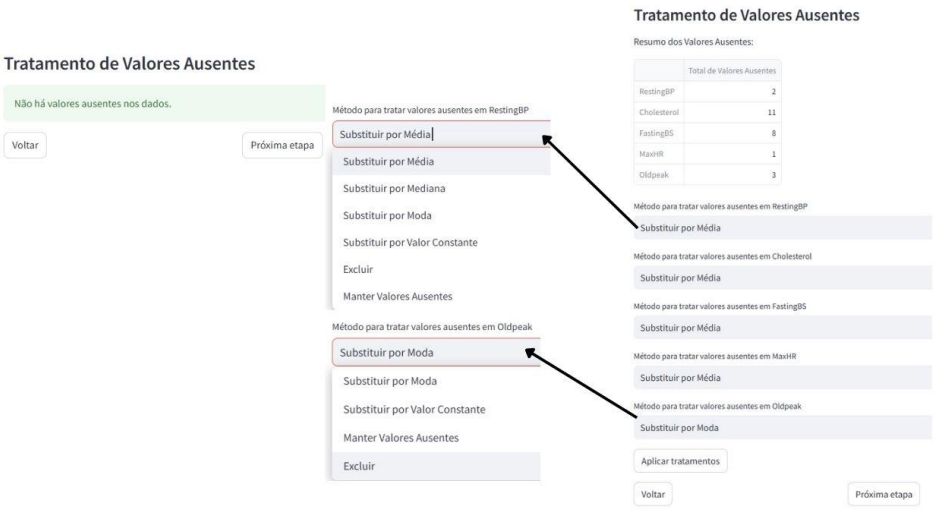
Para colunas numéricas, as opções de tratamento são:

- Substituir por Média, Mediana, Moda ou Valor Constante
- Excluir linhas com valores ausentes
- Manter valores ausentes

Para colunas categóricas, as opções são:

- Substituir por Moda ou Valor Constante
- Excluir linhas com valores ausentes
- Manter valores ausentes

Caso deseje alterar a opção de tratamento, clique na seta para baixo e escolha outra opção. Após fazer a seleção, clique no botão "Aplicar Tratamento" para que a plataforma aplique automaticamente o tratamento escolhido.



8

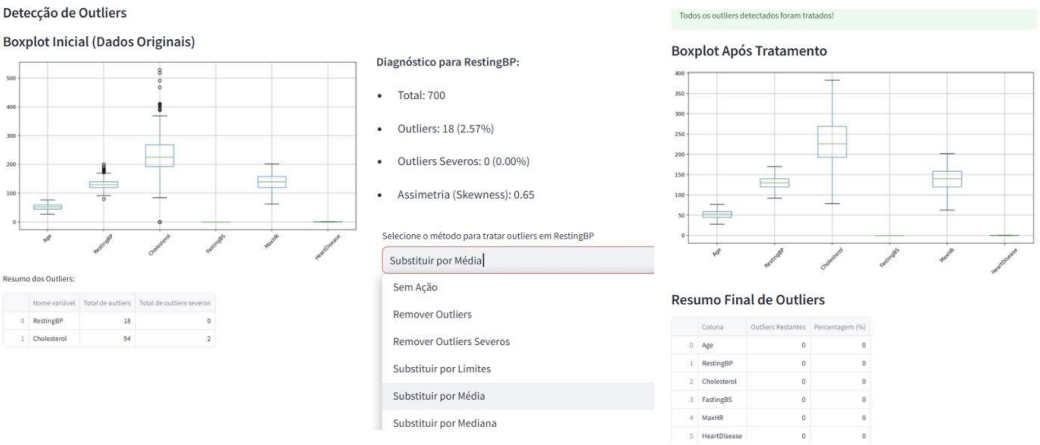
02.3 DETEÇÃO DE OUTLIERS

A plataforma deteta automaticamente os outliers e apresenta-os numa tabela e num boxplot inicial. Em seguida, são mostrados dados de diagnóstico para ajudar na tomada de decisão sobre como proceder.

A plataforma escolhe automaticamente a melhor forma de tratar os outliers, mas pode alterar essa escolha, se desejar. Para tal, basta clicar na seta para baixo e escolher outra opção. As opções de tratamento disponíveis são:

- Remover Outliers: Exclui todas as linhas com outliers.
- Remover Outliers Severos: Remove apenas os outliers mais extremos.
- Substituir por Média ou Mediana: Substitui os outliers pela média ou mediana da coluna.
- Substituir por Limites: Limita os valores dos outliers para um valor máximo ou mínimo.
- Sem Ação: Mantém os outliers nos dados, sem qualquer modificação.

Após escolher a opção desejada, clique no botão "Aplicar Tratamento" para que a plataforma aplique o tratamento selecionado. Depois de aplicar o tratamento, será exibida uma nova tabela e um novo boxplot, mostrando os resultados dos tratamentos aplicados aos outliers.

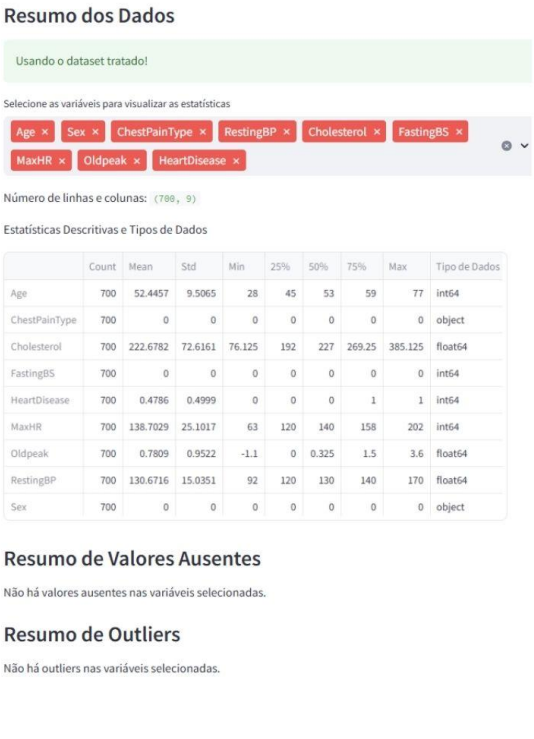


9

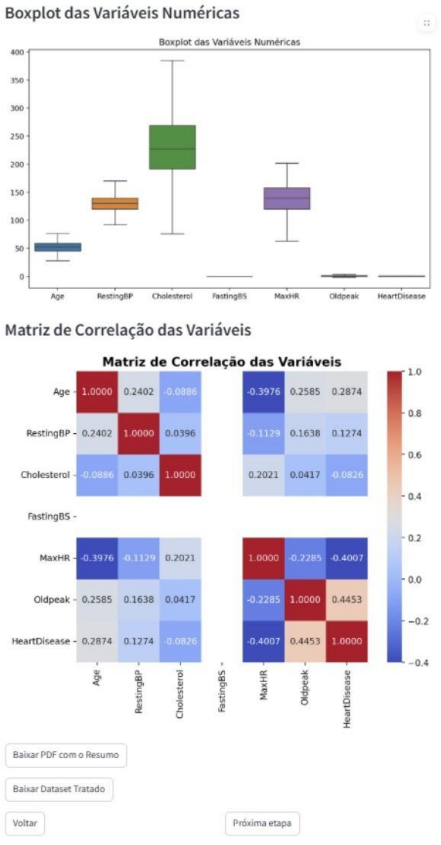
03 RESUMO DE DADOS

A plataforma gera um resumo completo dos dados carregados, que inclui informações cruciais sobre cada coluna do conjunto de dados. O resumo contém:

- Número de linhas e colunas.
- Tabela:
  - Estatísticas descritivas.
  - Tipos de dados de cada coluna.
- Resumo de valores ausentes.
- Resumo de outliers.
- Boxplot das variáveis numéricas.
- Matriz de correlação.
- Botão de download do Resumo em PDF
- Botão de download do *Dataset* Tratado



	Count	Mean	Std	Min	25%	50%	75%	Max	Tipo de Dados
Age	700	52.4457	9.5065	28	45	53	59	77	int64
ChestPainType	700	0	0	0	0	0	0	0	object
Cholesterol	700	222.6782	72.6161	76.125	192	227	269.25	385.125	float64
FastingBS	700	0	0	0	0	0	0	0	int64
HeartDisease	700	0.4786	0.4999	0	0	0	1	1	int64
MaxHR	700	138.7029	25.1017	63	120	140	158	202	int64
Oldpeak	700	0.7809	0.9522	-1.1	0	0.325	1.5	3.6	float64
RestingBP	700	130.6716	15.0351	92	120	130	140	170	float64
Sex	700	0	0	0	0	0	0	0	object



10

04 SELEÇÃO, TREINO E AVALIAÇÃO DE MODELOS

Esta etapa está dividida em 3 subetapas sendo ela:

- Modelos de Classificação
- Modelos de Regressão
- Modelos de Clustering

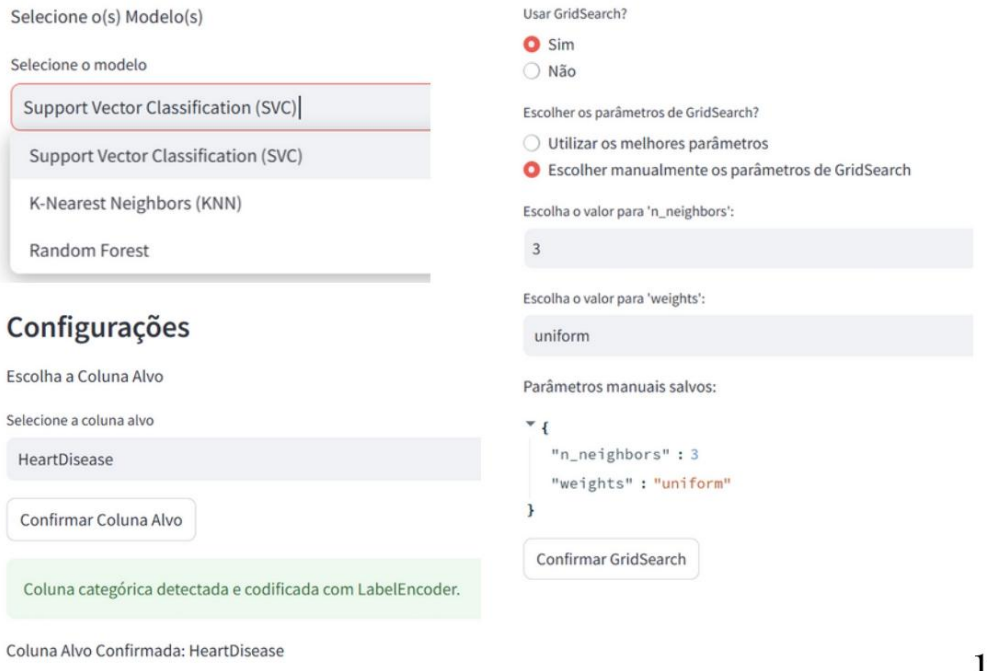
11

04.1 CLASSIFICAÇÃO

1. **Escolher o Modelo:** Selecione um dos seguintes modelos de classificação:
   - KNN (K-Nearest Neighbors)
   - SVC (Support Vector Classification)
   - Random Forest

2. Escolher a Coluna Alvo: Defina qual coluna será usada como variável alvo (a coluna que o modelo irá classificar).

3. Uso de GridSearch: Pode escolher usar ou não o gridsearch. Caso opte por usar escolha entre:
   - Deixar a plataforma escolher automaticamente os melhores parâmetros
   - Escolher manualmente os parâmetros, como:
     - KNN: n_neighbors, weights
     - SVC: C, kernel, e se for rbf, também gamma
     - Random Forest: n_estimators, max_depth



12

04.1 CLASSIFICAÇÃO

4. Escolher os Parâmetros de Validação: Selecione entre:

- **Divisão de Treino e Teste:** Defina a proporção de divisão.
- **Holdout:** Defina a proporção para treino e teste.

5. Treinar o Modelo: Clique no botão "Treinar Modelo". A plataforma treina o modelo e mostra:

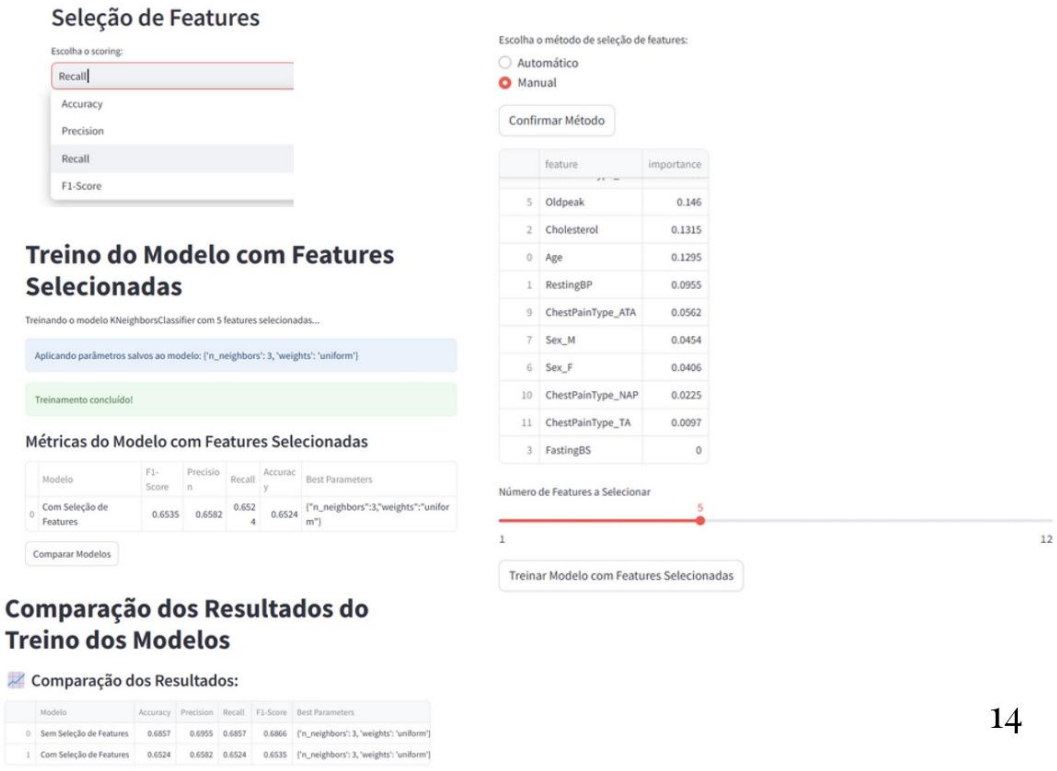
- **Resumo das Escolhas Feitas:** Detalhes do modelo, parâmetros e validação.
- **Tabela de Métricas:** Avaliação do desempenho do modelo.
- **Gráfico de Desempenho:** Gráfico com as métricas.

04.1 CLASSIFICAÇÃO

6. **Seleção de *Features*:**
   - **Escolher o *Scoring:*** Selecione a métrica (*Accuracy*, *Precision*, *Recall*, ou F1-*Score*).
   - **Escolher o Método de Seleção:**
     - **Automático:** A plataforma escolhe as melhores *features*.
     - **Manual:** Escolha o número de *features* usando o *slider*.
   - **Confirmar Seleção de *Features*:** Clique em "Confirmar Seleção de Features".

7. **Treino do Modelo com *Features* Selecionadas:** É apresentada a tabela das métricas com os resultados do treino com a seleção de *features*.

8. **Comparação dos Resultados:** A plataforma exibe:
   - *Features* antes e após a seleção.
   - Tabela comparando as métricas (com e sem seleção de *features*).
   - Gráfico de Comparação de métricas.



14

04.1 CLASSIFICAÇÃO

9. **Resumo Final:** Após clicar em "Seguir para Resumo Final":
   - **Configurações Utilizadas:** Mostra o tipo de modelo e o modelo selecionado.
   - **Informações dos Conjuntos de Dados:** Mostra nº de amostras de treino e teste, o nº de *features* originais e após seleção. Mostra tambem quais as *features* que foram selecionadas.
   - **Tabela de Comparação de Métricas:** Mostra as métricas para ambos os cenários (com e sem seleção de *features*).
   - **Gráfico Interativo para comparar as métricas.**
   - **Indicação do Melhor Modelo:** A plataforma mostra qual modelo teve melhor desempenho com base no critério escolhido.
   - **Interpretação das Métricas:** Análise do desempenho e sugestões de melhorias.
   - **Download do Melhor Modelo:** Baixe o modelo treinado (ex.: best_model_com_selecao_features.pkl).
   - **Download do Relatório PDF:** Baixe o relatório com as métricas e comparações.
   - **Concluir:** Após o download, clique em "Concluir" para finalizar o processo.

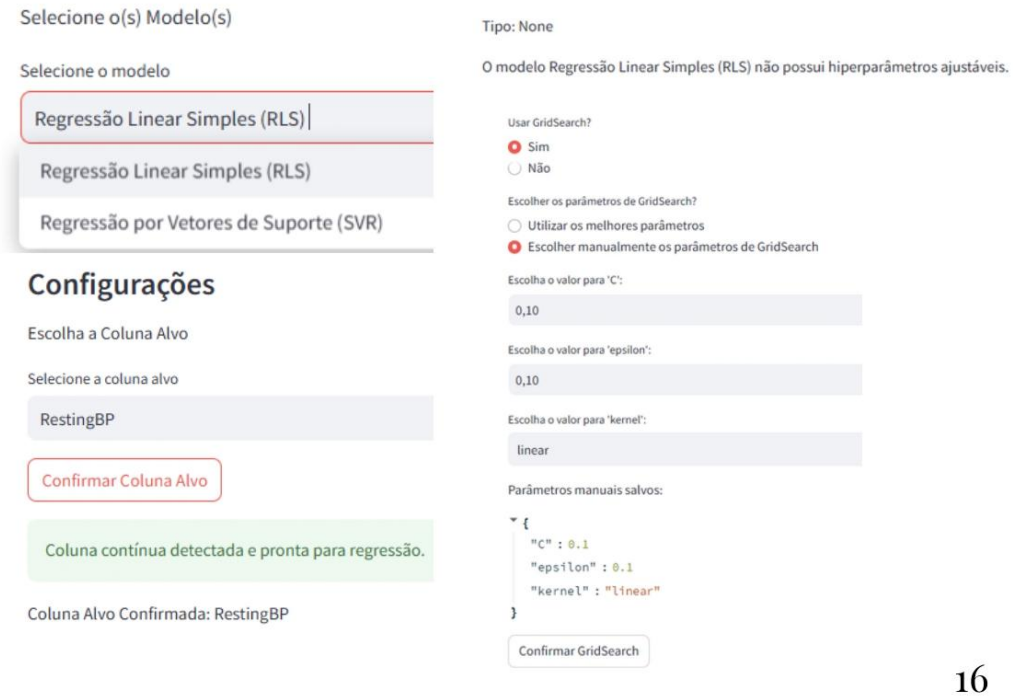
15

04.2 REGRESSÃO

1. **Escolher o Modelo:** Selecione um dos seguintes modelos de regressão:
   - Regressão Linear Simples (RLS)
   - Regressão por Vetores de Suporte (SVR)

2. Escolher a Coluna Alvo: Defina qual coluna será usada como variável alvo (a coluna que o modelo irá prever).

3. Uso de *GridSearch*: Pode escolher usar ou não o *gridsearch*. Caso opte por usar escolha entre:
   - Deixar a plataforma escolher automaticamente os melhores parâmetros
   - Escolher manualmente os parâmetros, como:
     - SVR: C, epsilon e Kernel.
     - RLS: Não há parâmetros adicionais, pois é um modelo linear simples que depende diretamente das variáveis independentes.



16

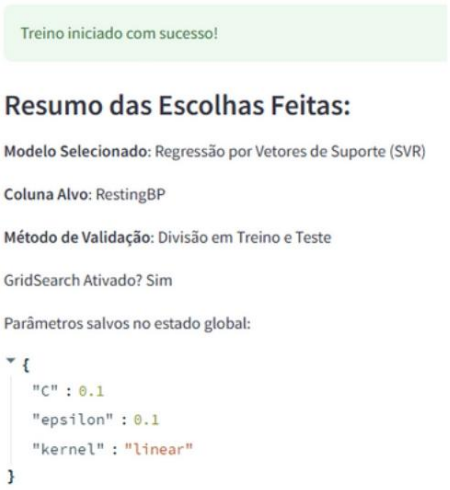
04.2 REGRESSÃO

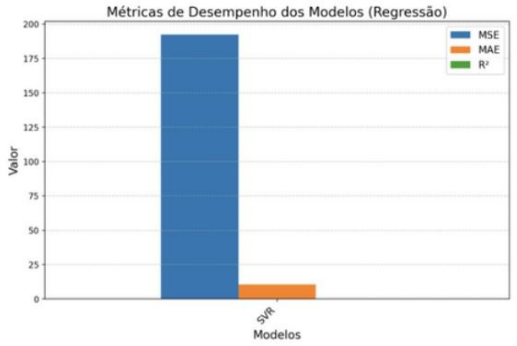
4. Escolher os Parâmetros de Validação: Selecione entre:

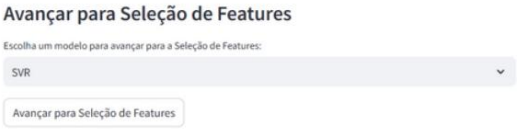
- **Divisão de Treino e Teste:** Defina a proporção de divisão.
- **Holdout:** Defina a proporção para treino e teste.

5. Treinar o Modelo: Clique no botão "Treinar Modelo". A plataforma treina o modelo e mostra:

- **Resumo das Escolhas Feitas:** Detalhes do modelo, parâmetros e validação.
- **Tabela de Métricas:** Avaliação do desempenho do modelo.
- **Gráfico de Desempenho:** Gráfico com as métricas.







Métricas do modelo treinado:

	Modelo	R^2	MAE	MSE	Best Parameters
0	SVR	0.0922	10.6772	192.2583	{'C': 0.1, 'epsilon': 0.1, 'kernel': 'linear'}

17

04.2 REGRESSÃO

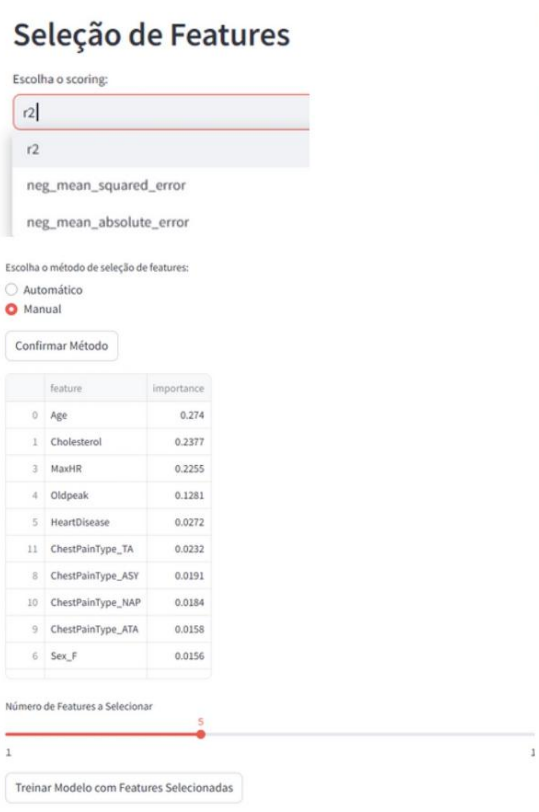
6. Seleção de *Features*:

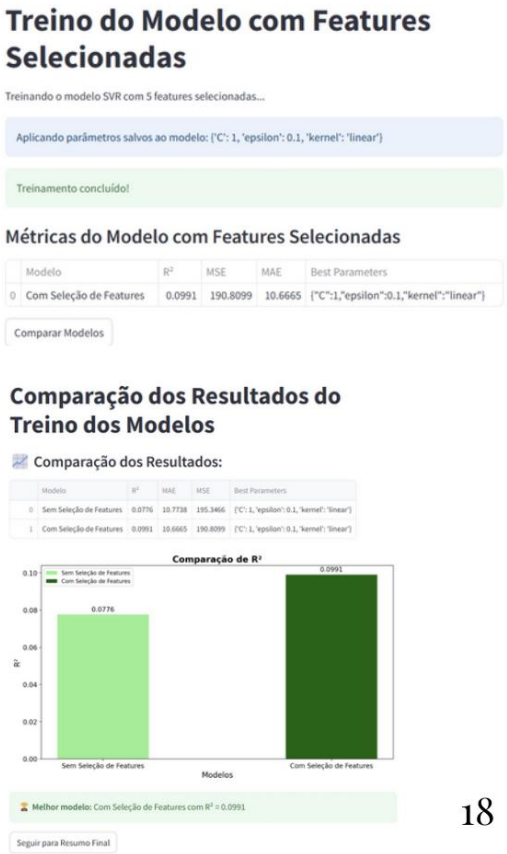
- **Escolher o *Scoring:*** Selecione a métrica (R^2, MAE e MSE).
- **Escolher o Método de Seleção:**
  - **Automático:** A plataforma escolhe as melhores *features*.
  - **Manual:** Escolha o número de *features* usando o *slider*.
- **Confirmar Seleção de *Features*:** Clique em "Confirmar Seleção de Features".

7. Treino do Modelo com *Features* Selecionadas: É apresentada a tabela das métricas com os resultados do treino com a seleção de *features*.

8. Comparação dos Resultados: A plataforma exibe:

- *Features* antes e após a seleção.
- Tabela comparando as métricas (com e sem seleção de features).
- Gráfico de Comparação de métricas.

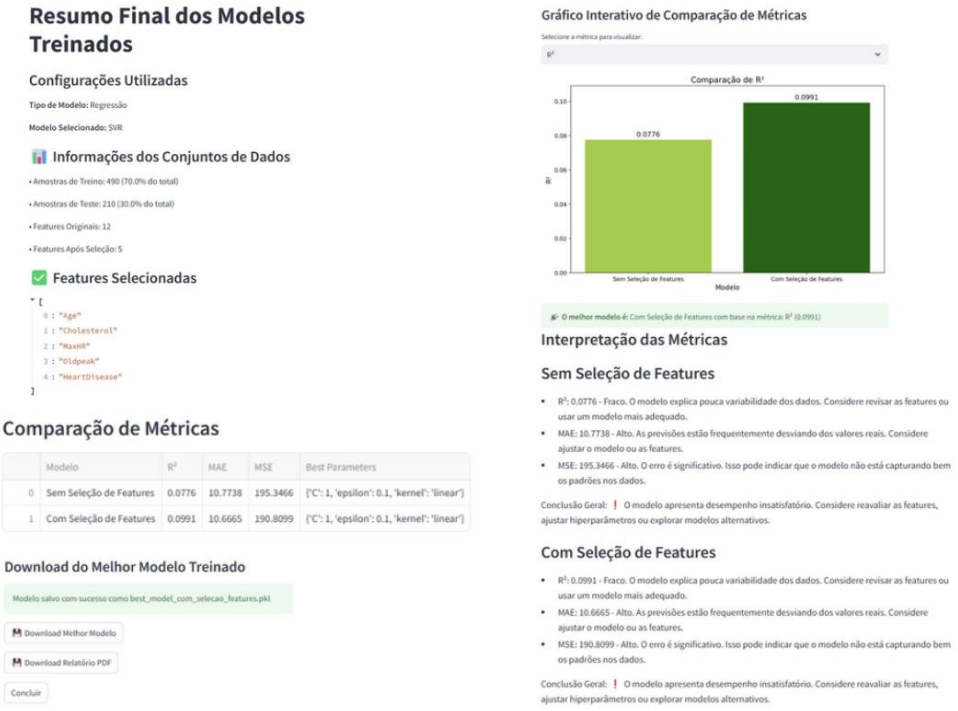




18

04.2 REGRESSÃO

9. **Resumo Final:** Após clicar em "Seguir para Resumo Final":
   - **Configurações Utilizadas:** Mostra o tipo de modelo e o modelo selecionado.
   - Informações dos Conjuntos de Dados: Mostra nº de amostras de treino e teste, o nº de features originais e após seleção. Mostra tambem quais as features que foram selecionadas.
   - **Tabela de Comparação de Métricas:** Mostra as métricas para ambos os cenários (com e sem seleção de *features*).
   - **Gráfico Interativo para comparar as métricas.**
   - **Indicação do Melhor Modelo:** A plataforma mostra qual modelo teve melhor desempenho com base no critério escolhido.
   - **Interpretação das Métricas:** Análise do desempenho e sugestões de melhorias.
   - **Download do Melhor Modelo:** Baixe o modelo treinado (ex.: best_model_com_selecao_features.pkl).
   - **Download do Relatório PDF:** Baixe o relatório com as métricas e comparações.
   - **Concluir:** Após o download, clique em "Concluir" para finalizar o processo.



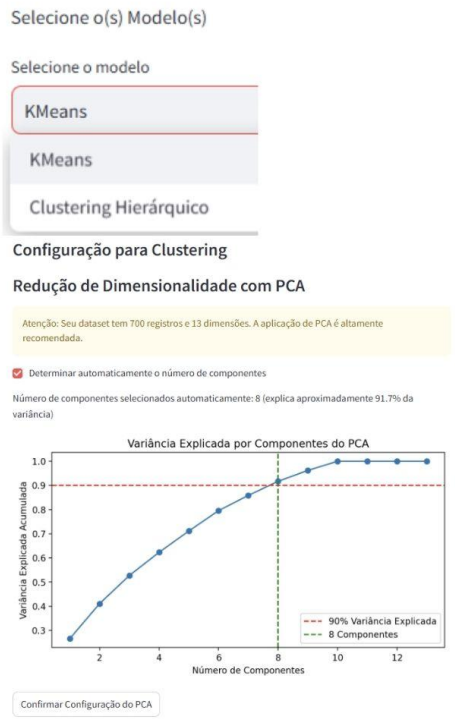
19

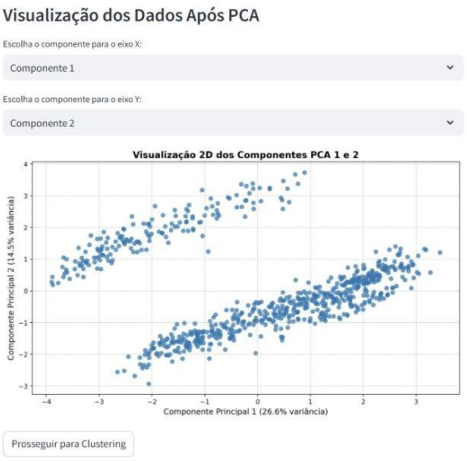
04.3 CLUSTERING

1. **Escolher o Modelo:** Selecione um dos seguintes modelos de clustering:
   - K-Means
   - Clustering Hierárquico

2. Escolher a Coluna Alvo: Ao contrário dos modelos de classificação e regressão, o clustering não exige uma coluna alvo.

3. Redução de Dimensionalidade com PCA: A plataforma permite determinar automaticamente ou manualmente o nº de componentes. Por defeito está selecionado automaticamente. Para escolher manualmente basta tirar o pisco da opção. Depois de confirmar a configuração do PCA é possível visualizar os dados após o PCA.





20

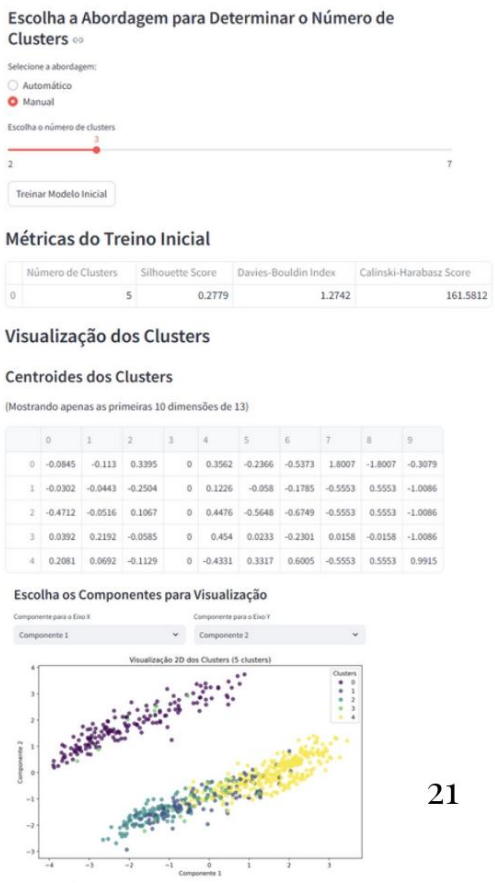
04.3 CLUSTERING

4. Determinação do Melhor Número de Clusters: A plataforma oferece uma análise detalhada das métricas para diferentes valores de K, permitindo visualizar suas variações antes de escolher a abordagem ideal. É possível também usar amostragem dos dados, o que acelera a análise em *datasets* grandes e pesados. Existem duas opções para definir o número de clusters:

- Deixar a plataforma escolher automaticamente
- Escolher manualmente o nº de K.

5. Treinar o Modelo: Clique no botão "Treinar Modelo Inicial". A plataforma treina o modelo e mostra uma tabela de Métricas para a avaliação do desempenho do modelo.

Após o treino inicial, a plataforma exibe uma caixa para escolher a próxima ação. Pode optar por re-treinar o modelo ou finalizar a análise.

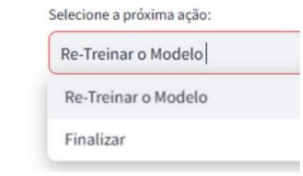


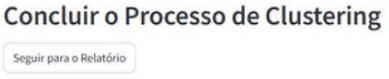
21

04.3 CLUSTERING

5. **Re-treino:** Após o treino inicial, pode optar por re-treinar o modelo com um novo número de clusters. Pode escolher novamente entre as abordagens automática ou manual para definir o número de clusters. Após escolher o novo valor, clique em "Treinar Novamente" para gerar as novas métricas.
Após o re-treino, as métricas do modelo treinado serão exibidas, permitindo comparar o desempenho com os resultados do treino inicial. Em seguida, pode clicar em "Seguir para Relatório" para avançar para a última etapa.

Nota:
Se optar por determinar automaticamente o valor de K em ambos os treinos, o resultado será semelhante, pois o valor de K será definido no primeiro treino. No entanto, é recomendado que, após escolher a opção automática no primeiro treino, selecione manualmente o valor de K no segundo treino.





22

04.3 CLUSTERING

8. **Relatório Final:** Após clicar em "Seguir para Relatório":
   - **Modelo utilizado:** Mostra o modelo selecionado.
   - **Número de Componentes PCA:** Mostra o nº de componentes PCA selecionados.
   - **Tabela de Comparação de Métricas:** Mostra as métricas para ambos os cenários.
   - **Gráfico Interativo para comparar as métricas.**
   - **Indicação do Melhor Modelo:** A plataforma mostra qual modelo teve melhor desempenho com base no Silhouette Score.
   - **Interpretação das Métricas:** Análise do desempenho do modelo.
   - **Download do Melhor Modelo:** Baixe o modelo treinado (ex.: melhor_modelo_treino_inicial.pkl).
   - **Download do Relatório PDF:** Baixe o relatório com as métricas e comparações.
   - **Concluir:** Após o download, clique em "Concluir" para finalizar o processo.

BIBLIOGRAFIA DE IMAGENS

- https://cdn-icons-png.flaticon.com/256/762/762724.png
- https://br.pinterest.com/pin/3359404534618250 42/ do corpo
- https://bizfunction.com/wp-content/uploads/2024/06/personalized-learning-icon.png
- https://cdn-icons-png.flaticon.com/512/7152/7152429.png
- https://cdn-icons-png.flaticon.com/256/944/944053.png
- https://static.vecteezy.com/ti/vetor-gratis/p1/14807189-icone-de-linha-de-analise-de-cluster-vetor.jpg
- https://medium.com/@anjunittur123/ols-vs-panel-ols-when-and-why-to-use-each-for-regression-analysis-534b275bd27b
- https://www.shutterstock.com/image-vector/classification-icon-vector-logotype-260nw-2271103469.jpg
- https://media2.dev.to/dynamic/image/width=800,height=,fit=scale-down,gravity=auto,format=auto/https://dev-to-uploads.s3.amazonaws.com/uploads/articles/39amhypqs5uowiozu3eb.png
- https://cdn-icons-png.flaticon.com/512/12635/12635967.png
- https://cdn-icons-png.freepik.com/512/7069/7069551.png
- https://cdno.iconfinder.com/data/icons/upload-download-files/128/file_xlsx_excel_document_upload-27-512.png
- https://static.vecteezy.com/ti/vetor-gratis/t1/28644017-json-arquivo-formato-linha-icone-gratis-vetor.jpg
- https://cdn-icons-png.freepik.com/512/11180/11180881.png
- https://miro.medium.com/v2/resize:fit:548/1*Yi22FxBWaVpQh4OF9CYOYA.png
- https://files.merlinapps.es/s3/almada/icons/services/9398eb7a-03b3-4546-a283-f0084fff5e59-65e1eac53b2b6.png
- https://www.kindpng.com/picc/m/483-4831825_arrow-clicking-mose-cursor-icon-hd-png-download.png
- https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcSUOogF_aOpROXp1kIleHYNyyKwwDCIZ571_92UkamPAkP8VSl3

BIBLIOGRAFIA DE IMAGENS

- https://static.wixstatic.com/media/1a4542_03535797107d483abc8b5dba42130742~mv2.png/v1/fill/w_473,h_468,al_c,lg_1,q_85/1a4542_03535797107d483abc8b5dba42130742~mv2.png https://cdn-icons-png.flaticon.com/512/2857/2857379.png https://cdn-icons-png.freepik.com/512/7440/7440395.png https://www.shutterstock.com/image-vector/cluster-analysis-icon-vector-image-260nw-2212563301.jpg
- https://banner2.cleanpng.com/20180604/hlk/aa9nb70qi.webp
- https://thumbs.dreamstime.com/b/m%C3%A30-que-faz-vetor-do-%C3%A2%E2%82%AC-sinal-%C3%ADcone-da-sele%C3%A7%C3%A30--136574601.jpg
- https://cdn-icons-png.flaticon.com/256/11744/11744373.png
- https://encrypted-tbn1.gstatic.com/images?q=tbn:ANd9GcSkfIicBPb2rShFNndB7mtJzJm4gA28twjtY30wsD7cC_PAzmDj
- https://cdn-icons-png.flaticon.com/512/9850/9850870.png
- https://cdn-icons-png.flaticon.com/512/5524/5524445.png
- https://thumbs.dreamstime.com/b/missing-data-icon-302522434.jpg
- https://media.istockphoto.com/id/1189314840/pt/vetorial/data-science-banner-web-icon-for-computer-science-and-insight-ai-big-data-algorithm.jpg?s=170667a&w=0&k=20&c=NE7axInU06qYV8lgDYZtnqnUZq8WfPhn2sSeJWv4K90=

Anexo B - Código para validação em ambiente *Python*

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import LabelEncoder
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LinearRegression
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, mean_absolute_error, mean_squared_error, r2_score
import joblib

# Configurações
RANDOM_STATE = 42
TEST_SIZE = 0.3

def load_data(file_path, delimiter=','):
    """Carrega os dados do arquivo especificado."""
    data = pd.read_csv(file_path, delimiter=delimiter)
    return data

def prepare_data(data, target_column):
    """Prepara os dados para treino."""
    X = data.drop(columns=[target_column])
    y = data[target_column]

    if y.dtype == 'object' or len(y.unique()) <= 10:
        problem_type = 'classification'
        le = LabelEncoder()
        y = le.fit_transform(y)
    else:
        problem_type = 'regression'

    X = pd.get_dummies(X)
    imputer = SimpleImputer(strategy='median')
    X = pd.DataFrame(imputer.fit_transform(X), columns=X.columns)

    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=TEST_SIZE, random_state=RANDOM_STATE
    )

    return X_train, X_test, y_train, y_test, problem_type

def train_model(X_train, X_test, y_train, y_test, model_name, scoring):
    """Treina um modelo escolhido pelo utilizador com a métrica de scoring
definida."""
    if model_name == "KNN":
        model_params = {'n_neighbors': [5], 'weights': ['uniform']}
        model = KNeighborsClassifier()
        grid_search = GridSearchCV(model, model_params, cv=5, scoring=scoring,
n_jobs=-1)
        grid_search.fit(X_train, y_train)
        best_model = grid_search.best_estimator_
        y_pred = best_model.predict(X_test)
```

```
        metrics = {
            "Accuracy": round(accuracy_score(y_test, y_pred), 4),
            "Precision": round(precision_score(y_test, y_pred,
average='weighted', zero_division=0), 4),
            "Recall": round(recall_score(y_test, y_pred, average='weighted',
zero_division=0), 4),
            "F1-Score": round(f1_score(y_test, y_pred, average='weighted',
zero_division=0), 4),
            "Best Parameters": str(grid_search.best_params_)
        }
    elif model_name == "LinearRegression":
        model = LinearRegression()
        model.fit(X_train, y_train)
        y_pred = model.predict(X_test)

        metrics = {
            "R² Score": round(r2_score(y_test, y_pred), 4),
            "MAE": round(mean_absolute_error(y_test, y_pred), 4),
            "MSE": round(mean_squared_error(y_test, y_pred), 4),
            "Best Parameters": "N/A"
        }
        best_model = model
    else:
        raise ValueError("Modelo não suportado. Escolha entre 'KNN' ou
'LinearRegression'.")

    return best_model, metrics

def run_validation(file_path, target_column, model_name, scoring, n_features=5,
output_file="validation_results.txt"):
    data = load_data(file_path)
    X_train, X_test, y_train, y_test, problem_type = prepare_data(data,
target_column)

    output_lines = []
    output_lines.append("\n=== Informações do Conjunto de Dados ===")
    output_lines.append(f"Amostras de Treino: {X_train.shape[0]}
({(X_train.shape[0] / len(data)) * 100:.2f}%)")
    output_lines.append(f"Amostras de Teste: {X_test.shape[0]}
({(X_test.shape[0] / len(data)) * 100:.2f}%)")

    # Avaliação SEM Seleção de Features
    model_before, metrics_before = train_model(X_train, X_test, y_train, y_test,
model_name, scoring)

    # Avaliação COM Seleção de Features (Baseado na importância das features)
    base_model = RandomForestClassifier(random_state=RANDOM_STATE) if
problem_type == "classification" else
RandomForestRegressor(random_state=RANDOM_STATE)
    base_model.fit(X_train, y_train)
    feature_importances = pd.Series(base_model.feature_importances_,
index=X_train.columns).nlargest(n_features)
    selected_features = feature_importances.index.tolist()
```

```
    output_lines.append("\nFeatures Selecionadas:")
    output_lines.append(str(selected_features))

    X_train_selected = X_train[selected_features]
    X_test_selected = X_test[selected_features]

    model_after, metrics_after = train_model(X_train_selected, X_test_selected,
y_train, y_test, model_name, scoring)

    # Criar DataFrame no formato correto
    comparison_df = pd.DataFrame({
        "Modelo": ["Sem Seleção de Features", "Com Seleção de Features"],
        **{metric: [metrics_before[metric], metrics_after[metric]] for metric in
metrics_before.keys()}
    })

    output_lines.append("\n=== Comparação de Métricas ===")
    output_lines.append(comparison_df.to_string(index=False))

    # Salvar resultados em arquivo
    with open(output_file, "w") as f:
        f.write("\n".join(output_lines))

    print("\nResultados salvos em:", output_file)

    # Salvar modelo final
    joblib.dump(model_after, "best_model.pkl")

    return comparison_df

if __name__ == "__main__":
    file_path = r"C:\\Users\\brunaso\\Downloads\\Bruna\\dataset_tratado (1).csv"
    target_column = "Preço" #"Classificação de Limpeza"
    model_name = "LinearRegression"  # Escolha entre "KNN" ou "LinearRegression"
    scoring = "r2"  # Escolha a métrica desejada, ex: "accuracy", "precision",
"r2"

    comparison_df = run_validation(file_path, target_column, model_name,
scoring, n_features=5)
```