CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
(CEFET-MG)

Vinicius Ferreira Almeida de Oliveira

DESENVOLVIMENTO DE MÉTODO DE CLASSIFICAÇÃO DA QUALIDADE DE ÁGUAS DE IRRIGAÇÃO VIA APRENDIZADO DE MÁQUINA

Belo Horizonte (MG)
2025

Vinicius Ferreira Almeida de Oliveira

DESENVOLVIMENTO DE MÉTODO DE CLASSIFICAÇÃO DA QUALIDADE DE ÁGUAS DE IRRIGAÇÃO VIA APRENDIZADO DE MÁQUINA

Trabalho de conclusão de curso apresentado como requisito parcial para a obtenção do título de Bacharel em Química Tecnológica.
Orientador: Prof. Dr. Cleverson Fernando Garcia.
Coorientadora: Bel. Christiane Aparecida Santos Silva.

CEFET-MG

Belo Horizonte (MG)

2025

DESENVOLVIMENTO DE MÉTODO DE CLASSIFICAÇÃO DA QUALIDADE DE ÁGUAS DE IRRIGAÇÃO VIA APRENDIZADO DE MÁQUINA

Trabalho de conclusão de curso do Bacharelado em
Química Tecnológica
CEFET-MG

Belo Horizonte, 8 de julho de 2025

Prof. Dr. Cleverson Fernando Garcia
(Orientador – CEFET-MG)

Bel. Christiane Aparecida Santos Silva
(Coorientadora – CEFET-MG)

Profª. Dra. Patricia Sueli de Rezende
(avaliadora – CEFET-MG)

Prof. Dr. Breno Rodrigues Lamaghere Galvão
(avaliador – CEFET-MG)

RESUMO

OLIVEIRA, V. F. A; SILVA, C. A. S.; GARCIA, C. F. . Desenvolvimento de Método de Classificação da Qualidade de Águas de Irrigação Via Aprendizado de Máquina

A água de irrigação desempenha papel fundamental na produtividade agrícola, sendo responsável por cerca de 70% do consumo global de água doce. Sua qualidade influencia diretamente a saúde das culturas, a estrutura dos solos e a eficiência dos sistemas de irrigação, exigindo monitoramento constante e avaliação precisa. A crescente complexidade da análise com o aumento do volume dos dados físico-químicos envolvidos, como condutividade elétrica, pH, sólidos dissolvidos e razão de adsorção de sódio, exige o uso de ferramentas computacionais avançadas para interpretar grandes volumes de informações ambientais. Este trabalho propõe o desenvolvimento de um método de classificação da qualidade da água para irrigação com base em algoritmos de aprendizado de máquina, com ênfase nos modelos de Árvores de Decisão e Florestas Randômicas. A base de dados, composta por 368 amostras contendo parâmetros físico-químicos de análise, foi obtida na plataforma Kaggle e tratada no ambiente Google Colab com o uso das bibliotecas Python Pandas, Scikit-learn, NumPy e Seaborn. A análise estatística descritiva, essencial na formação do químico tecnológico, evidenciou a presença de dados assimétricos e discrepantes em parâmetros como pH, condutividade elétrica e íons cloreto e sódio, reforçando a necessidade de técnicas robustas para modelagem preditiva. Após a padronização das variáveis, os dados foram divididos em conjuntos de treinamento e teste. Os modelos foram avaliados com base nas métricas médias de acurácia (99%), precisão (98%), revocação (97%) e F1-score (98%), calculadas ao longo de 100 execuções independentes. O modelo de Árvore de Decisão obteve o melhor desempenho geral, mesmo mantendo os *outliers* nos dados. O presente estudo evidenciou, ainda, a destacada importância da variável sólidos totais dissolvidos no processo de classificação. Por fim, os resultados demonstraram o potencial da Inteligência Artificial como ferramenta estratégica para a gestão de recursos hídricos e sugerem sua aplicação em contextos com maior variabilidade regional e maior complexidade de interação entre os parâmetros.

Palavras-chave: qualidade da água; irrigação; Aprendizado de Máquina; Árvore de Decisão; Floresta Randômica; classificação.

SUMÁRIO

1. INTRODUÇÃO

A água é um recurso natural essencial para a manutenção da vida e das atividades humanas. Na agricultura, sua importância é ainda mais evidente, sendo utilizada em grande escala para garantir a produção de alimentos e a segurança alimentar global. A irrigação, prática agrícola que consiste na aplicação controlada de água ao solo para suprir as necessidades hídricas das plantas, representa aproximadamente 70% do consumo mundial de água doce (ITABORAHY et al., 2004). Diante da escassez crescente de recursos hídricos e da intensificação das atividades agrícolas, tornasse fundamental garantir não apenas a disponibilidade, mas também a qualidade da água utilizada para irrigação.

As chamadas águas de irrigação podem ter origem em rios, lagos, poços subterrâneos ou mesmo em efluentes tratados. Entretanto, sua qualidade pode variar consideravelmente em função da fonte e das atividades antrópicas na região, apresentando diferentes teores de sais dissolvidos, matéria orgânica, metais e microrganismos. O uso de águas inadequadas para irrigação pode causar salinização e sodificação do solo, redução na produtividade agrícola, toxicidade para as plantas, além de representar riscos à saúde pública e ao meio ambiente (ALMEIDA, 2010; SANTOS, 2000).

Nesse contexto, torna-se imprescindível a avaliação sistemática da qualidade da água para irrigação, com base em parâmetros físico-químicos estabelecidos por órgãos reguladores, como pH, condutividade elétrica, sólidos totais dissolvidos (TDS), razão de adsorção de sódio (RAS), cloretos, nitratos, dureza total, entre outros (AYERS; WESTCOT, 1987).

Entretanto, a interpretação conjunta de diversos parâmetros analíticos pode se tornar um desafio, sobretudo em bancos de dados extensos, nos quais podem existir correlações complexas, comportamentos assimétricos e a presença de *outliers*. Nesses casos, o uso de ferramentas estatísticas e computacionais torna-se necessário para uma análise eficaz e para a geração de diagnósticos ambientais confiáveis. O profissional de Química Tecnológica, com formação sólida em análise de dados experimentais, química analítica e sistemas de controle, está habilitado a atuar diretamente nessa interface entre a ciência dos dados e as questões ambientais.

Com o avanço da ciência da computação, técnicas de Inteligência Artificial (IA) têm sido cada vez mais aplicadas no campo ambiental. Entre elas, destaca-se o Aprendizado de Máquina, que permite que algoritmos sejam treinados para reconhecer padrões e tomar decisões com base em grandes conjuntos de dados (MITCHELL, 1997; MURPHY, 2012). Diferentes modelos vêm sendo aplicados para tarefas de classificação e predição, sendo as Árvores de Decisão, Florestas Randômicas e Redes Neurais Artificiais os mais utilizados em

estudos envolvendo monitoramento da qualidade da água e análise de risco ambiental (GOODFELLOW; BENGIO; COURVILLE, 2016; BIAU; SCORNET, 2015).

Esses algoritmos são capazes de lidar com relações não lineares, selecionar automaticamente os atributos mais relevantes e gerar modelos com elevada acurácia, desde que adequadamente calibrados. Além disso, permitem avaliações comparativas entre diferentes conjuntos de dados e métodos, auxiliando na identificação de padrões que seriam difíceis de observar por métodos tradicionais. Aplicações bem-sucedidas de Aprendizado de Máquina na área ambiental incluem desde a classificação de corpos d'água, passando pela previsão de índices de qualidade, até a detecção de anomalias em parâmetros físico-químicos (LI et al., 2022; PANG et al., 2021).

Diante da relevância do tema e da necessidade de otimização de métodos de avaliação da qualidade da água, este trabalho tem por objetivo desenvolver um método de classificação da qualidade de águas de irrigação com base em parâmetros físico-químicos, utilizando algoritmos de aprendizado de máquina. A base de dados, composta por 368 amostras, foi tratada por meio de técnicas estatísticas descritivas e padronização, e os modelos foram avaliados com base em métricas como acurácia, precisão, revocação e F1-score. A análise comparativa entre os modelos busca identificar aquele com maior robustez e aplicabilidade prática, contribuindo para a gestão eficiente dos recursos hídricos, a promoção da sustentabilidade na agricultura e o fortalecimento da interdisciplinaridade entre a Química Tecnológica e a Ciência de Dados.

2. REVISÃO BIBLIOGRÁFICA

2.1. Águas de Irrigação

A agricultura é a principal consumidora de água no mundo, utilizando cerca de 69% do volume global disponível em rios, lagos e aquíferos (ITABORAHY et al., 2004). Dentro deste contexto, a irrigação é a atividade que mais demanda recursos hídricos, sendo essencial para o aumento da produtividade agrícola e a segurança alimentar. No entanto, a qualidade da água utilizada para irrigação desempenha papel crucial para o sucesso das práticas agrícolas, influenciando diretamente a saúde das culturas, a sustentabilidade do solo e a eficiência dos sistemas de irrigação (ALMEIDA, 2010).

A alta demanda de água para o agronegócio, segundo Belizario (2014), está ligada ao fato das plantas perderem muita água por evapotranspiração, exigindo consideráveis quantidades para a reposição hídrica. Embora seja a atividade econômica que mais consome água, a suficiência mundial de produção de alimentos, depende da agricultura irrigada, porque

embora ocupe somente 16% da área produtiva, é responsável por 40% da produção total de alimentos. Assim, medidas de controle da quantidade de água aplicada através de técnicas de irrigação que melhoram a eficiência da aplicação de água e são imprescindíveis para reduzir os impactos sobre os recursos hídricos. Também são importantes, antes da implantação de novos perímetros irrigados, estabelecer limites sustentáveis para a utilização da água, por meio da determinação dos aspectos quantitativos dos recursos hídricos (vazões médias e mínimas) e o monitoramento da sazonalidade da disponibilidade hídrica ao longo do ano.

A escolha de fontes de água de má qualidade pode levar a problemas como a salinização dos solos, a contaminação dos produtos agrícolas e o comprometimento da saúde dos trabalhadores rurais. Assim, torna-se imprescindível avaliar e monitorar a qualidade da água antes de sua utilização na irrigação (SANTOS, 2000).

Normalmente, o monitoramento ambiental de uma grande quantidade de parâmetros traz dificuldades na interpretação dos resultados exigindo conhecimentos na área. A partir daí, faz-se necessário o uso de abordagens, como a estatística multivariada, que permitem uma redução dos dados e uma interpretação de diversos constituintes de forma conjunta. A análise de componentes principais, por exemplo, permite a redução de variáveis por meio de critérios objetivos, sendo de grande impacto no estudo ambiental de bacias e corpos hídricos (MAGALHÃES, 2022).

Estudos demonstram que, nos casos em que a condutividade elétrica da água excede 1500 µS/cm, há redução no consumo hídrico e no rendimento de culturas como alface (GERVÁSIO; CARVALHO; SANTANA, 2021). Em regiões semiáridas, o uso de águas salobras ou tratadas exige estratégias específicas — como manejo em pulsos e uso de culturas tolerantes — para evitar a lixiviação de sais e prejuízos à saúde do solo (MENDES FILHO et al., 2024).

Recentemente, estudos têm evidenciado que técnicas de Inteligência Artificial oferecem resultados promissores na análise ambiental. Simão & Pécora Jr. (2020) realizaram uma revisão sobre o uso de IA no tratamento de efluentes, destacando aplicações de redes neurais nebulosas em processos biológicos. Já Rubio et al. (2019) desenvolveram um protótipo sensor com redes neurais para identificação automática de contaminantes ambientais.

A qualidade da água para irrigação é avaliada por meio de parâmetros físicos, químicos e biológicos, que ajudam a identificar potenciais riscos à produtividade agrícola e ao meio ambiente. Esses parâmetros são regulamentados por órgãos nacionais e internacionais, como o

Conselho Nacional do Meio Ambiente (CONAMA) no Brasil e a Organização das Nações Unidas para a Alimentação e a Agricultura (FAO) em nível global.

Segundo CONAMA (2005), são diversos os parâmetros físico-químicos utilizados na avaliação da qualidade das águas de irrigação, entre eles:

- Turbidez: indica a presença de partículas suspensas na água, podendo obstruir sistemas de irrigação por aspersão e gotejamento;
- Temperatura: afeta a solubilidade de gases e nutrientes na água, além de influenciar o metabolismo das plantas;
- Cor: geralmente associada à presença de matéria orgânica dissolvida ou outros compostos químicos;
- Sólidos Suspensos Totais (SST): refletem a quantidade de partículas sólidas presentes na água, que podem reduzir a eficiência dos sistemas de irrigação e afetar a infiltração no solo.

Os parâmetros químicos, por sua vez, também são importantes na avaliação da qualidade, destacando-se:

- pH: deve estar entre 6,0 e 9,0 para evitar problemas de toxicidade e corrosão nos equipamentos de irrigação (ALMEIDA, 2010);
- Condutividade Elétrica (CE): mede a concentração de sais dissolvidos na água. Valores altos indicam risco de salinidade e comprometem a absorção de nutrientes pelas plantas (CONAMA, 2005);
- Relação de Absorção de Sódio (RAS): avalia o risco de sodificação do solo, que afeta sua permeabilidade e estrutura (CONAMA, 2005);
- Carbonatos e Bicarbonatos: em concentrações elevadas, podem precipitar cálcio e magnésio, favorecendo a sodificação (CONAMA, 2005);
- Elementos Tóxicos: íons como sódio (Na^{+}) e, cloreto (Cl^{-}), além do boro, quando associados à matéria orgânica e em concentrações excessivas podem causar toxicidade às plantas além de íons de metálicos como cádmio (Cd^{2+}), chumbo (Pb^{2+}) e mercúrio (Hg^{2+}), podem se acumular, no solo e causar impactos negativos à saúde humana e ambiental (CONAMA, 2005).

Por fim, segundo o CONAMA (2005), os parâmetros biológicos também afetam sistematicamente a qualidade das águas de irrigação, sendo eles:

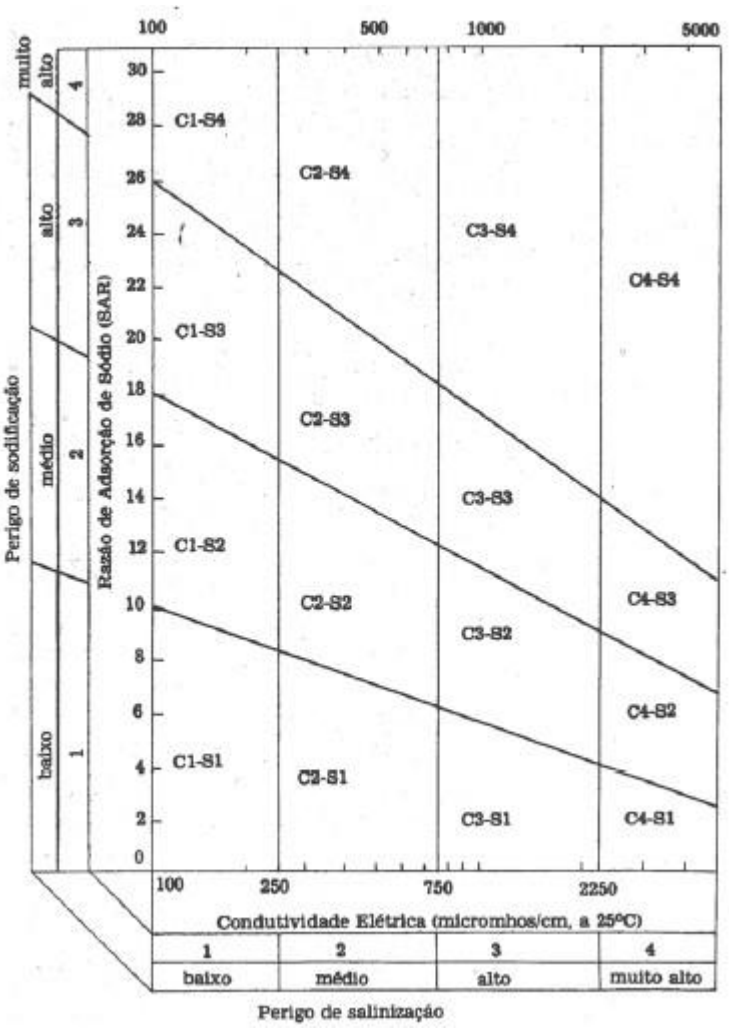
- Coliformes Totais e Fecais: indicadores de contaminação por matéria orgânica e microrganismos patogênicos;
- Organismos Patogênicos: podem incluir bactérias, vírus e protozoários, representando riscos à saúde humana e animal.

A Classificação da qualidade da água para irrigação pode ser feita de acordo com protocolos distintos, tendo destaque o sistema do Laboratório de Salinidade dos Estados Unidos da América, que classifica a água com base na condutividade elétrica (CE) e na relação de absorção de sódio (RAS), organizando-as em diferentes categorias:

- Classes de Salinidade de acordo com a condutividade elétrica:
  - C1: baixa salinidade (CE ≤ 250 μS/cm). Adequada para todos os tipos de solos e culturas;
  - C2: salinidade moderada (CE entre 250 e 750 μS/cm). Requer manejo de lixiviação para evitar acúmulo de sais;
  - C3: alta salinidade (CE entre 750 e 2250 μS/cm). Necessita de drenagem eficiente e práticas de controle de salinidade;
  - C4: muito alta salinidade (CE > 2250 μS/cm). Uso restrito a condições especiais com culturas tolerantes a sais.
- Classes de Sodicidade de acordo com a relação de absorção de sódio:
  - S1: baixo risco de sódio trocável (RAS < 10);
  - S2: risco moderado de sódio (RAS entre 10 e 18);
  - S3: alto risco de sódio (RAS entre 18 e 26);
  - S4: muito alto risco de sódio (RAS > 26).

A Classificação é construída com base no diagrama da Figura 1. Para utilizar o diagrama deve se conhecer a condutividade elétrica e as concentrações de sódio, cálcio e magnésio necessários para o cálculo da RAS. A condutividade elétrica é obtida utilizando um condutívimetro e o teor de sódio utilizando fotômetro de chama (ALMEIDA, 2010).

Figura 1: Diagrama de classificação de águas



Fonte: ALMEIDA (2010)

A FAO, por sua vez considera a qualidade da água em relação à salinidade, toxicidade por íons específicos e problemas de infiltração (ALMEIDA, 2010). Essa classificação divide as águas em três categorias de restrição:

I. Sem restrição: utilização direta em qualquer cultura;

II. Restrição leve a moderada: requer práticas de manejo, como drenagem e lixiviação;

III. Restrição severa: uso limitado a condições específicas, com cultivos tolerantes a sais.

Esse sistema utiliza da Tabela 1 o estabelecimento das classificações se baseando nos resultados analíticos da composição química, permitindo ajustes no manejo e numa melhor compreensão dos efeitos da água no solo (ALMEIDA, 2010).

Tabela 1: Diretrizes para interpretação da qualidade das águas para irrigação.

Problema Potencial	Unidade	Grau de Restrição de Uso		
		Nenhum	Ligeiro a Moderado	Severo
Salinidade (afeta a disponibilidade de água para o cultivo)[2]				
CE_{ai} ou	dS m^{-1}	<0,7	0.7 – 0.3	>3.0
TSD	mg L^{-1}	<450	450 – 2000	>2000
Infiltração (reduz a infiltração; avaliar usando a RAS° e a CE_{ai})[3]				
RAS° = 0 - 3 e CE_{ai} =	dS m^{-1}	>0.7	0.7 – 0.2	<0.2
= 3 – 6 e CE_{ai} =	dS m^{-1}	>1.2	1.2 – 0.3	<0.3
= 6 – 12 e CE_{ai} =	dS m^{-1}	>1.9	1.9 – 0.5	<0.5
= 12 – 20 e CE_{ai} =	dS m^{-1}	>2.9	2.9 – 1.3	<1.3
= 20 – 40 e CE_{ai} =	dS m^{-1}	>5.0	5.0 – 2.9	<2.9
Toxicidade de Íons Específicos (afeta cultivos sensíveis)				
Sódio (Na^+)[4]				
Irrigação por superfície	RAS°	<3	3 – 9	>9
Irrigação por aspersão	meq L^{-1}	<3	>3	
Cloro (Cl^-)[4]				
Irrigação por superfície	meq L^{-1}	<4	4.0 – 10	>10
Irrigação por aspersão	meq L^{-1}	<3	>3	
Boro (B)[5]	mgr L^{-1}	<0.7	0.7 – 3.0	>3
Vários (afeta cultivos sensíveis)				
Nitrogênio (NO_3 N)[6]	mgr L^{-1}	<5	5.0 – 30	>30
Bicarbonato (HCO_3) (aspersão foliar unicamente)	meq L^{-1}	<1.5	1.5 – 8.5	>8.5
pH	Amplitude Normal 6,5 – 8,4			

Fonte: ALMEIDA, 2010

Por sua vez, o Conselho Nacional do Meio Ambiente regulamenta a qualidade da água, no Brasil, com base na Resolução nº 357/2005 (CONAMA, 2005), classificando-a em três categorias principais:

- Água Doce: salinidade ≤ 0,5‰;
- Água Salobra: salinidade entre 0,5‰ e 30‰;
- Água Salina: salinidade ≥ 30‰.

Ao considerar especificamente os parâmetros vinculados à qualidade, a resolução CONAMA 357/2005 estabelece limites máximo por classe, como evidenciado, a seguir:

- pH: entre 6,0 e 9,0;
- Condutividade elétrica: valores dependem do tipo de cultura e solo, mas não devem exceder os níveis que causem salinização significativa;
- Demanda Bioquímica de Oxigênio (DBO): indicador da matéria orgânica biodegradável presente;
- Óleos e graxas: devem ser mínimos para evitar contaminação dos solos;

- Coliformes Termotolerantes: máximo de 1.000 NMP/100 mL para águas destinadas à irrigação de hortaliças consumidas cruas.

Além disso, a resolução reconhece variações regionais, permitindo ajustes para condições naturais específicas, como em Sete Lagoas (MG), onde a alta concentração de cálcio e magnésio resulta em maior dureza da água.

2.2. Inteligência Artificial

A Inteligência Artificial (IA) é uma área interdisciplinar da ciência da computação que busca desenvolver sistemas capazes de realizar tarefas que exigem habilidades cognitivas humanas, como raciocínio, aprendizado, percepção e tomada de decisão. Suas aplicações abrangem setores variados, como saúde, transporte, educação e entretenimento, promovendo mudanças significativas na forma como interagimos com a tecnologia (RUSSELL; NORVIG, 2021).

Os primeiros passos da IA podem ser rastreados até meados do século XX, com a formalização de conceitos matemáticos que possibilitaram o desenvolvimento de máquinas computacionais avançadas. Segundo Turing (1950), a questão central para a IA seria: "Máquinas podem pensar?". Essa reflexão abriu espaço para o desenvolvimento de sistemas que simulam o raciocínio humano, inicialmente em áreas como resolução de problemas e jogos.

Ao longo das décadas, o campo evoluiu, dividindo-se em subáreas como aprendizado de máquina, redes neurais, visão computacional e processamento de linguagem natural (GOODFELLOW; BENGIO; COURVILLE, 2016). Os avanços mais recentes incluem também o desenvolvimento de algoritmos de aprendizado profundo (*deep learning*), que são capazes de reconhecer padrões complexos em grandes volumes de dados (LECUN; BENGIO; HINTON, 2015). Esses progressos foram fortemente impulsionados pelo aumento da capacidade computacional, pela disponibilidade de big data e pelo aprimoramento de técnicas de modelagem matemática (JORDAN; MITCHELL, 2015).

A IA pode ser classificada em três categorias principais: IA fraca (ou estreita), projetada para realizar tarefas específicas; IA geral, capaz de executar qualquer tarefa cognitiva que um ser humano possa realizar; e a IA superinteligente, um conceito ainda teórico que ultrapassaria a capacidade humana em todos os aspectos cognitivos (BOSTROM, 2014).

Atualmente, a IA é amplamente aplicada em diversos setores. Na saúde, sistemas baseados em IA auxiliam no diagnóstico precoce de doenças, análise de imagens médicas e desenvolvimento de medicamentos personalizados (TOPOL, 2019). Na indústria, robôs

inteligentes otimizam processos produtivos e reduzem custos operacionais (BRYNJOLFSSON; MCAFEE, 2017). Além disso, na área de transporte, os veículos autônomos representam uma das mais inovadoras utilizações da IA prometendo maior segurança e eficiência no trânsito. No setor agrícola, a IA tem sido utilizada para previsão de safras, controle de irrigação, detecção de pragas e, mais recentemente, no monitoramento da qualidade da água utilizada na produção (SINGH et al., 2020). Apesar de seus avanços, a IA levanta questões éticas e sociais que precisam ser enfrentadas. O impacto no mercado de trabalho, a possível amplificação de vieses algorítmicos e o uso indevido de sistemas de vigilância são algumas das preocupações mais discutidas. De acordo com Floridi et al. (2018), o desenvolvimento responsável da IA deve considerar princípios como transparência, justiça e respeito à privacidade, de modo a garantir que os benefícios da tecnologia sejam amplamente distribuídos.

2.3. Aprendizado de Máquina

O Aprendizado de Máquina é um ramo da Inteligência Artificial que se concentra no desenvolvimento de algoritmos e sistemas capazes de aprender padrões a partir de dados, melhorando seu desempenho ao longo do tempo sem a necessidade de programação explícita. Essa área vem desempenhando um papel central no avanço da IA e é amplamente utilizada em tarefas como classificação, regressão, reconhecimento de padrões e previsão (MITCHELL, 1997).

O conceito de Aprendizado de Máquina surgiu na década de 1950, com as primeiras tentativas de criar algoritmos que aprendessem por meio de dados. Um marco inicial foi o trabalho de Arthur Samuel, que cunhou o termo "Aprendizado de Máquina" ao desenvolver um programa de computador capaz de jogar damas e melhorar seu desempenho com a prática (SAMUEL, 1959).

Desde então, o campo evoluiu significativamente, impulsionado por avanços em estatística, matemática e ciência da computação. Na década de 1980, surgiram as redes neurais artificiais, que imitavam o funcionamento do cérebro humano, mas só ganharam maior relevância nos anos 2010, com o crescimento do aprendizado profundo (do inglês *deep learning*), viabilizado pelo aumento do poder computacional e pela disponibilidade de grandes conjuntos de dados (GOODFELLOW; BENGIO; COURVILLE, 2016).

No contexto do aprendizado supervisionado, os algoritmos podem ser utilizados para regressão — quando o objetivo é prever um valor numérico contínuo (como concentração de

íons, por exemplo) — ou para classificação, quando o objetivo é atribuir uma amostra a uma entre várias categorias possíveis, como no caso da classificação da qualidade da água.

Os métodos de Aprendizado de Máquina são tradicionalmente classificados em três categorias principais:

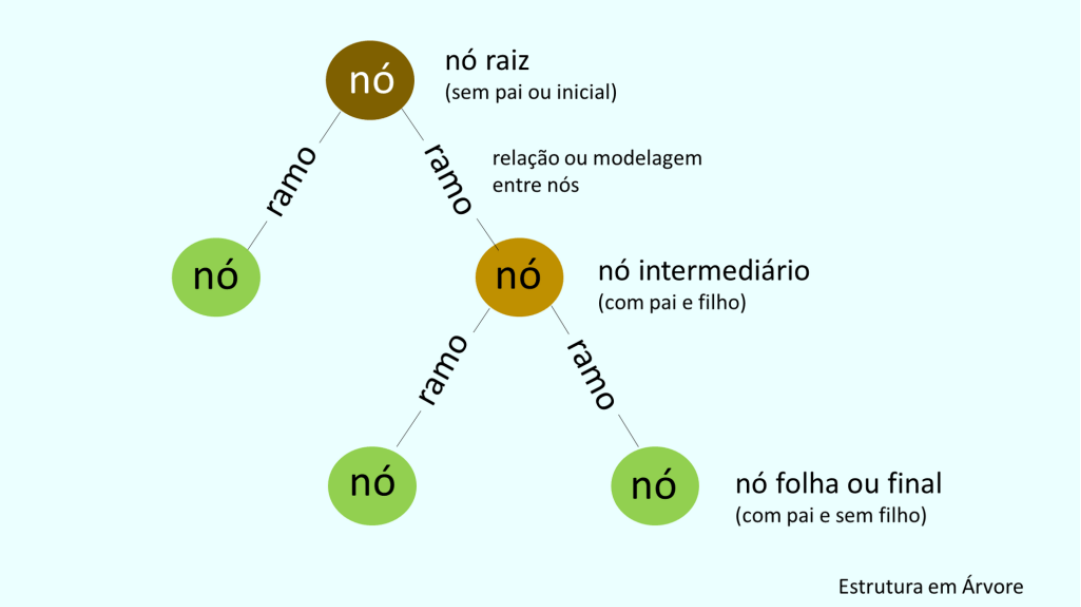
I. Aprendizado Supervisionado: algoritmos que aprendem a partir de dados rotulados, ou seja, pares de entrada e saída, para prever resultados futuros (HASTIE; TIBSHIRANI; FRIEDMAN, 2009). Os modelos desse método são: Regressão Linear, Regressão Logística, Máquinas de Vetores de Suporte (SVM), Árvores de Decisão, Florestas Randômicas, k-Nearest Neighbors (k-NN) e Redes Neurais Artificiais;
II. Aprendizado Não Supervisionado: técnicas que identificam padrões ou estruturas ocultas em dados não rotulados. Seus modelos são: Clusterização, Redução de Dimensionalidade e Modelos Generativos;
III. Aprendizado por Reforço: abordagem baseada em agentes que aprendem interagindo com um ambiente para maximizar uma recompensa cumulativa (SUTTON; BARTO, 2018). Sendo composto por: Q-Learning, Deep Q-Networks (DQN), Algoritmos Baseados em Políticas e Métodos de Monte Carlo.

Dentre os modelos utilizados na classificação de amostras estão as Árvores de Decisão, as Florestas Randômicas e as Redes Neurais Artificiais.

2.3.1. Árvores de Decisão

As Árvores de Decisão (Figura 2) se destacam por oferecerem modelos interpretáveis e visualmente intuitivos, permitindo que o processo de decisão seja compreendido de forma clara. Essas árvores estruturam os dados de maneira hierárquica, facilitando a identificação de regras e critérios que conduzem à classificação ou predição (SHALEV-SHOWTZ; BEN-DAVID, 2014). As árvores de decisão são representações gráficas que organizam decisões de maneira hierárquica. Cada nó interno da árvore contém uma pergunta ou condição baseada em um atributo do conjunto de dados, enquanto cada folha representa uma decisão final ou uma classe predita. Durante o treinamento, algoritmos como ID3, CART e C4.5 selecionam os atributos que melhor separam os dados, utilizando medidas como o ganho de informação ou a impureza (HASTIE; TIBSHIRANI; FRIEDMAN, 2009).

Figura 2: Imagem ilustrativa de um modelo de árvore de decisão



Fonte: IUNES, 2023

Segundo GAO (2021) a construção de uma árvore de decisão segue um conjunto de etapas para ser plenamente estruturada, evidenciadas a seguir:

I. Seleção de Atributos: Nesta etapa, o algoritmo analisa os atributos disponíveis e escolhe aquele que resulta na maior redução de impureza ou maior ganho de informação, determinando a melhor forma de particionar o conjunto de dados.

II. Divisão Recursiva: Após a escolha do atributo ideal, o conjunto é particionado em subconjuntos que se tornam nós da árvore. Esse processo é repetido de forma recursiva em cada nó, até que se atinja um critério de parada, como a homogeneidade dos dados ou um número mínimo de instâncias por nó.

III. Poda: Para evitar o sobreajuste (*overfitting*), ou seja, o ajuste excessivo do modelo aos dados de treinamento, são aplicadas técnicas de poda. Estas técnicas removem ramos que apresentam baixa relevância preditiva, contribuindo para uma melhor generalização do modelo.

IV. Avaliação do Modelo: A árvore construída é avaliada com base em um conjunto de dados de teste ou por meio de validação cruzada, verificando se o modelo mantém sua eficácia ao classificar ou predizer novos dados.

Gao (2021) relata, ainda, que por se tratar de um modelo simples essas árvores possuem vantagens e algumas limitações sendo:

Vantagens:

- Interpretabilidade: A estrutura em árvore permite uma visualização clara das decisões e dos critérios aplicados;
- Flexibilidade: Podem ser utilizadas tanto para problemas de classificação quanto de regressão;
- Eficiência: Geralmente, apresentam um tempo de treinamento reduzido e são fáceis de implementar.

Limitações:

- Sensibilidade aos Dados: Pequenas variações nos dados podem resultar em árvores muito diferentes;
- Sobreajuste: Árvores muito profundas podem se ajustar excessivamente aos dados de treinamento, prejudicando a generalização. Métodos como poda e junção de árvores (Ex.: Florestas Randômicas) são empregados para mitigar esse problema.

Durante o treinamento, as Árvores de Decisão utilizam critérios de divisão baseados em medidas de impureza, como:

- Índice de Gini: mede a frequência com que um elemento escolhido aleatoriamente seria incorretamente classificado, com base na distribuição das classes;
- Entropia (ou Ganho de Informação): baseada na teoria da informação, mede a incerteza de um sistema. A entropia é máxima quando os dados estão uniformemente distribuídos entre as classes e mínima quando pertencem a uma única classe.

Essas medidas ajudam o algoritmo a selecionar os atributos mais informativos para particionar os dados nos nós da árvore.

2.3.2. Floresta Randômica

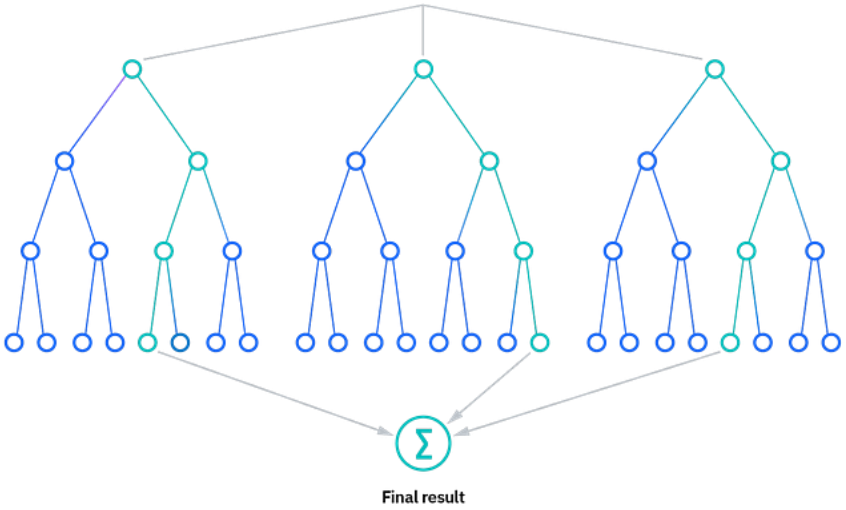
Ao considerar a Floresta Randômica (Figura 3), a ideia central é construir um conjunto (*ensemble*) de árvores de decisão na qual cada árvore é treinada com um subconjunto dos dados e um subconjunto aleatório dos atributos (BREIMAN, 2001). Essa abordagem ajuda a reduzir o risco de sobreajuste (*overfitting*) e melhora a precisão da predição. As Florestas Randômicas funcionam da seguinte forma:

I. Criação do Ensemble de Árvores: São geradas diversas árvores de decisão, em que para cada árvore uma amostra aleatória (com reposição, método *bootstrap*) do conjunto de dados é selecionada. Além disso, em cada divisão (nó) da árvore, um subconjunto aleatório de atributos é considerado para determinar a melhor divisão.

II. Treinamento de Cada Árvore: Cada árvore é treinada de forma independente utilizando os dados e atributos selecionados aleatoriamente. O algoritmo de construção da árvore segue os mesmos princípios dos métodos tradicionais, mas a aleatoriedade introduzida aumenta a diversidade entre as árvores.

III. Agregação dos Resultados: Para realizar a predição em novos dados, as saídas de todas as árvores são combinadas. No caso de problemas de classificação, é utilizada a votação majoritária; para regressão, a média dos valores preditos é calculada. Essa agregação tende a reduzir a variância e a melhorar a estabilidade do modelo.

Figura 3: Imagem ilustrativa de um modelo de floresta randômica



Fonte: SEGATTO, 2023

O treinamento de um modelo Floresta Randômica envolve as seguintes etapas:

I. Pré-processamento e Amostragem: Após a preparação dos dados, geram-se amostras *bootstrap* (com reposição) do conjunto original, criando subconjuntos do mesmo tamanho para cada árvore do *ensemble* (CHAOUDHARY, 2021);

II. Construção das Árvores: Em cada árvore, e para cada nó, seleciona-se aleatoriamente um subconjunto dos atributos (mtry). A divisão ótima no nó é determinada com base em critérios estatísticos como o índice de Gini ou a entropia, construindo a árvore sem poda (MOOREN, 2025; BIAU; SCORNET, 2015);

III. Agregação e Validação: As previsões das árvores são combinadas — voto majoritário para classificação e média para regressão. A validação pode ser feita por validação cruzada, por um conjunto de teste separado ou pela estimativa de erro

out-of-bag (OOB), usando os exemplos que não entraram nas respectivas amostras *bootstrap* (CHAOUDHARY, 2021; LOUPE, 2014);

IV. Otimização dos Parâmetros: Ajustam-se hiperparâmetros como o número de árvores (ntree), o número de atributos por divisão (mtry) e a profundidade máxima da árvore, buscando melhorar a performance do modelo e evitar redundância entre árvores (CURTIN, 2025; LOUPE, 2014).

Assim como as árvores de decisão, as Florestas Randômicas também possuem vantagens e limitações:

Vantagens:

- Redução do Sobreajuste: A combinação de múltiplas árvores treinadas com amostragem aleatória reduz significativamente a variância e evita o *overfitting* (MOOREN, 2025; BIAU; SCORNET, 2015);
- Robustez: O modelo é menos sensível a ruídos e variações no conjunto de dados devido à agregação do *ensemble* (LOUPE, 2014);
- Escalabilidade: Funciona bem em dados volumosos e com alta dimensionalidade, e pode ser facilmente paralelizado (CURTIN, 2025).

Limitações:

- Interpretabilidade reduzida: A junção de muitas árvores dificulta a compreensão global da lógica de decisão, mesmo que cada árvore individual seja interpretável (LOUPE, 2014);
- Custo Computacional: Treinar centenas ou milhares de árvores demanda tempo e recursos elevados, especialmente com conjuntos de dados grandes (CURTIN, 2025).

2.3.3. Redes Neurais Artificiais

As Redes Neurais Artificiais (RNAs) constituem um dos pilares do aprendizado de máquina e da inteligência artificial. Inspiradas no funcionamento do cérebro humano, essas redes são compostas por unidades de processamento interconectadas – os neurônios artificiais – que, organizados em camadas, permitem aprender padrões complexos a partir de dados. Essa técnica tem se destacado em tarefas como reconhecimento de imagem, processamento de linguagem natural e previsão de séries temporais (GOODFELLOW; BENGIO; COURVILLE, 2016).

O funcionamento das RNAs baseia-se na interação entre neurônios organizados em camadas,

uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída. Cada neurônio realiza uma operação que consiste na soma ponderada dos sinais recebidos, seguida pela aplicação de uma função de ativação não linear, a fim de introduzir complexidade na modelagem dos dados. Esse processamento distribuído permite que a rede aprenda representações hierárquicas, essenciais para capturar padrões complexos (NIELSEN, 2015).

Ainda segundo Nilsen (2015) para realizar o treinamento de uma rede neural artificial devese seguir as seguintes etapas:

1. Propagação Direta: Os dados de entrada são propagados pela rede, passando por cada camada e produzindo uma saída. Durante essa fase, cada neurônio calcula sua saída com base nos pesos atuais;
2. Retropropagação do Erro: Após a obtenção da saída, calcula-se o erro (diferença entre a saída prevista e a saída desejada) por meio de uma função de custo. Esse erro é então propagado de volta pela rede, ajustando os pesos dos neurônios pelo emprego de algoritmos de otimização, como o Gradiente Descendente. Esse processo iterativo minimiza a função de custo e melhora a capacidade de generalização da rede.

Assim como os modelos anteriores, as RNAs possuem uma série de vantagens e limitações em sua construção.

Vantagens:

- Capacidade de Modelagem: Permitem capturar relações não lineares e padrões complexos;
- Adaptabilidade: São capazes de aprender com grandes volumes de dados e se adaptar a diferentes tipos de problemas;
- Versatilidade: Aplicáveis a uma ampla gama de tarefas, desde classificação e regressão até processamento de linguagem natural e visão computacional.

Limitações:

- Requisitos Computacionais: O treinamento de redes profundas pode demandar alto poder computacional e tempo de processamento.;
- Ajuste de Hiperparâmetros: A definição adequada de parâmetros (como taxa de aprendizagem, número de camadas e neurônios) é crucial e, muitas vezes, desafiadora;
- Interpretabilidade: Apesar dos avanços, redes profundas ainda são consideradas "caixaspretas", dificultando a explicação completa de suas decisões.

O Aprendizado de Máquina tem sido amplamente aplicado em diversos setores. Na área da saúde, algoritmos ajudam no diagnóstico de doenças, como detecção precoce de câncer por meio de imagens médicas (TOPOL, 2019). Na indústria, é usado para prever manutenção de

máquinas e otimizar processos produtivos (BRYNJOLFSSON; MCAFEE, 2017). No setor financeiro, sistemas de aprendizado de máquina são empregados para prever riscos de crédito e detectar fraudes, na indústria de alimentos no controle de qualidade de bebidas (CORTEZ, 2009).

Embora o Aprendizado de Máquina tenha avançado significativamente, ainda enfrenta desafios importantes. Questões como a necessidade de grandes volumes de dados de qualidade, o viés algorítmico e a dificuldade de interpretar modelos complexos, como redes neurais profundas, são temas de debate contínuo. Além disso, o impacto ético e social de sua aplicação, especialmente em áreas sensíveis, como reconhecimento facial, requer atenção.

2.4. Parâmetros de avaliação em Aprendizado de Máquina

Os parâmetros de avaliação em Aprendizado de Máquina desempenham um papel crucial no processo de validação e comparação de modelos. Eles permitem medir a eficácia de algoritmos ao realizar tarefas específicas, fornecendo subsídios para ajustes e melhorias. Além disso, a escolha adequada das métricas de avaliação é essencial para garantir que os modelos sejam generalizáveis e úteis em aplicações do mundo real (MURPHY, 2012).

A avaliação de modelos de aprendizado de máquina vai além de verificar o desempenho nos dados de treinamento. Ela se concentra na capacidade do modelo de generalizar para novos dados, minimizando erros e garantindo um desempenho confiável. Segundo Géron (2019), uma avaliação criteriosa ajuda a evitar problemas como *overfitting* e *underfitting*, que comprometem a eficácia do modelo em cenários reais.

Os parâmetros de avaliação variam de acordo com o tipo de problema abordado, como classificação, regressão ou aprendizado não supervisionado segundo MURPHY, 2012. A seguir, destacam-se as métricas mais utilizadas:

1. Métricas para Classificação
    - Acurácia: mede a proporção de previsões corretas em relação ao total de observações. Embora amplamente utilizada, não é indicada em conjuntos de dados desbalanceados;
    - Precisão (Precision): mede a fração de previsões positivas corretas, sendo útil em cenários em que os falsos positivos têm maior impacto;
    - Revocação (Recall): indica a fração de instâncias positivas corretamente identificadas pelo modelo;

- F1-Score: combina precisão e revocação em uma única métrica harmônica, ideal para problemas com classes desbalanceadas;
- Matriz de Confusão: proporciona uma visão completa do desempenho do modelo, contabilizando verdadeiros positivos, verdadeiros negativos, falsos positivos e falsos negativos (KAPLAN; HAENLEIN, 2019).

Calculados utilizando as equações a seguir:

$$AC = \frac{TP + TN}{TP + TN + FP + FN} \quad \text{(Eq.1)}$$

$$PR = \frac{TP}{\mathrm{TP} + \mathrm{FP}} \quad \text{(Eq.2)}$$

$$RV = \frac{TP}{TP + FN} \quad \text{(Eq.3)}$$

$$F1 = 2 * \frac{PR * RV}{PR + RV} \quad \text{(Eq.4)}$$

Sendo:

AC: Acurácia

PR: Precisão

RV: Revocação

F_1: F1-score

TP: Verdadeiro Positivos

TN: Verdadeiros Negativos

FP: Falsos Positivos

FN: Falsos Negativos

2. Métricas para Regressão

- Erro Quadrático Médio (MSE): penaliza erros maiores de forma mais severa ao elevar ao quadrado a diferença entre previsões e valores reais;
- Erro Absoluto Médio (MAE): mede o erro médio absoluto, sendo menos sensível a *outliers*;
- R^2 (Coeficiente de Determinação): avalia a proporção da variância nos dados explicada pelo modelo, sendo uma métrica amplamente usada para determinar a qualidade da regressão (GÉRON, 2019);

- Curvas de avaliação ROC e AUC: a curva ROC mede o desempenho do modelo ao comparar a taxa de verdadeiros positivos (TPR) e falsos positivos (FPR). A área sob a curva (AUC) oferece uma métrica global de desempenho;
- Curva Precision-Recall: especialmente útil em problemas com classes desbalanceadas, demonstra a relação entre precisão e revocação em diferentes *thresholds* (MURPHY, 2012).

Problemas com classes desbalanceadas, como detecção de fraudes e diagnóstico de doenças raras, representam desafios específicos para a avaliação de modelos. A simples utilização da acurácia pode ser enganosa, já que um modelo que prevê sempre a classe majoritária pode obter alta acurácia, mas baixo desempenho nas classes minoritárias. Técnicas como reamostragem e métricas específicas, como F1-score e AUC-ROC, são essenciais nesses contextos (CHOLLET, 2018).

Por fim, destaca-se o contínuo avanço na criação de métricas mais sofisticadas, como as usadas para explicar o comportamento de redes neurais profundas (SHAP e LIME) e para avaliar *fairness* em algoritmos. Essas métricas ajudam a mitigar problemas éticos e garantir maior confiabilidade em sistemas críticos, como os utilizados em saúde e justiça (MÜLLER; GUIDO, 2016).

3. METODOLOGIA

3.1. Base de dados

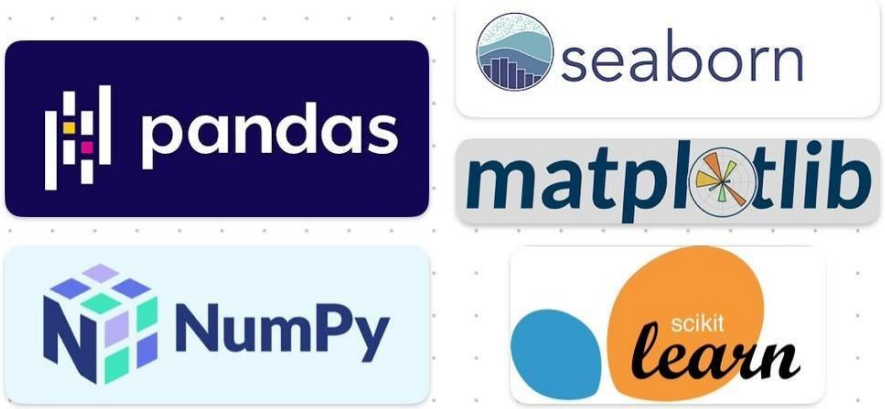
As planilhas utilizadas no presente trabalho são provenientes de dados físico-químicos de corpos de água da região de Tangala na Índia. Elas foram obtidas na plataforma de estudos em Aprendizado de Máquina Kaggle, possuindo, aproximadamente 368 amostras que foram analisadas para avaliação de qualidade e classificação. Destaca-se que os arquivos foram disponibilizados na plataforma nos anos de 2018, 2019 e 2020.

Entre todos os parâmetros físico-químicos, selecionou-se pH, sólidos totais dissolvidos e os teores dos íons bicarbonato, cloreto, nitrato, sulfato, sódio, potássio, cálcio e magnésio. A variável classe, por sua vez, foi estruturada com as classes de águas C2, C3 e C4, relacionadas à condutividade elétrica.

3.2. Tratamento dos dados

O tratamento dos dados foi realizado na plataforma Google Colab utilizando a linguagem Python para o processamento por meio de suas bibliotecas: Pandas, Numpy, Matplotlib, Seaborn e Scikit-Learn (Figura 4).

Figura 4: Logos das bibliotecas Python para análise de dados



Fonte: Nur, 2024

Inicialmente, o arquivo contendo os resultados das análises físico-químicos, de extensão ‘csv’, foi convertido em um *dataframe* para organização e visualização dos dados. Em seguida selecionou-se as variáveis relacionadas aos parâmetros físico-químicos, descartando-se as demais informações por não estarem relacionadas aos objetivos do presente trabalho.

O estudo de análises descritivas foi realizado por meio do cálculo de medidas de tendência central (média aritmética amostral e mediana amostral) e variabilidade dos dados (desvio-padrão amostral). A representação gráfica dos dados foi realizada utilizando boxplots.

Após o estudo descritivo, a planilha resultante foi dividida em base de dados dos atributos preditores e base de dados do atributo classe. Para tanto, criou-se novos *dataframes* fazendo uso do indexador função iloc.

A base de dados dos atributos preditores passou, ainda, por uma padronização dos dados (Equação 5), considerando a diversidade de unidades dos dados e consequente diferença entre as escalas. Para tanto, utilizou-se a função StandardScaler pertencente à biblioteca Scikit-learn.

$$Zi = (Xi - \bar{X})/s \qquad \text{(Eq. 5)}$$

Z_i: Valor padronizado

X_i: Dado individual

$\bar{X}$: Média aritmética

s: desvio-padrão

A base de dados do atributo classe, por sua vez, foi composta por uma única coluna apresentando a classificação de qualidade de cada amostra.

Após a referida estruturação dados, as bases de dados foram utilizadas para a elaboração dos conjuntos de dados de treinamento e de teste. Para tanto, utilizou-se a função train_test_split, relacionada à biblioteca Scikit-learn, garantindo 75% das amostras para o treinamento e 25% para o teste (GARCIA, 2022).

O conjunto de dados de treinamento e de teste foram utilizadas na criação de modelos de aprendizado de máquina de árvore de decisão, floresta randômica e de redes neurais artificiais, buscando-se aquele cuja eficiência de classificação fosse mais elevada. As métricas de avaliação utilizadas no processo comparativo foram acurácia de treinamento, acurácia de previsão, precisão, revocação e F1-score (GERON, 2019).

3.3. Aplicação dos modelos de Aprendizado de Máquina

Nesta etapa, foram aplicados três modelos de Aprendizado de Máquina — Árvore de Decisão, Floresta Randômica e Redes Neurais Artificiais — para a classificação da qualidade da água de irrigação. O objetivo foi comparar o desempenho dos modelos com base nas métricas previamente definidas, identificando aquele com melhor capacidade preditiva.

Após o tratamento e a padronização dos dados, estes foram divididos em conjuntos de treinamento (75%) e de teste (25%). Essa divisão aleatória foi realizada de forma independente em 100 execuções distintas, permitindo avaliar a variação natural nas métricas de desempenho. Em cada repetição, os modelos foram treinados com os dados de treinamento e avaliados com os dados de teste, e os resultados foram armazenados para análise posterior.

A seleção dos hiperparâmetros ideais foi realizada por meio de validação cruzada com busca em grade (GridSearchCV), a partir do conjunto de treinamento. Essa etapa garantiu que cada modelo fosse ajustado para maximizar seu desempenho preditivo antes da avaliação final. A título de exemplo, no caso da Árvore de Decisão, foram otimizados parâmetros como o critério de divisão (Gini ou entropia), a profundidade máxima da árvore, o número mínimo de amostras por divisão e por nó folha.

Finalizada a otimização, os modelos foram submetidos às 100 execuções independentes com os hiperparâmetros fixos. Os valores das métricas de avaliação foram então utilizados para calcular as médias aritméticas e os desvios-padrão. Esse procedimento teve como finalidade garantir maior robustez estatística e minimizar o impacto de uma única divisão aleatória dos dados sobre os resultados obtidos.

Para a comparação dos modelos foi utilizado o teste de Wilcoxson ($\alpha = 0,05$) relacionado a cada métrica, considerando cada classe.

4. RESULTADOS E DISCUSSÃO

4.1. Análise descritiva

A análise descritiva dos dados físico-químicos das amostras de águas de irrigação, considerando a base de dados estudada revelou importantes características relacionadas à qualidade das amostras de água e ao potencial de uso para fins agrícolas, especialmente irrigação. A Tabela 2 sintetiza os principais parâmetros mensurados em 368 amostras.

Tabela 2: Análise descritiva dos dados

Parâmetro	Média	Mediana	Mínimo	Máximo
pH	7,84	7,85	6,28	9,23
Condutividade Elétrica (E.C) (µS/cm)	1414,7	1174,5	261,0	9499,0
TDS (mg/L)	905,4	751,7	167,0	6079,4
Cloreto (Cl^-) (mg/L)	205,6	130,0	10,0	2480,0
Fluoreto (F^-) (mg/L)	1,02	0,84	0,04	7,7
Nitrato (NO_3^-) (mg/L)	80,5	39,7	0,1	1028,0
Sódio (Na^+) (mg/L)	124,3	86,0	6,0	748,1
SAR (Razão de Adsorção de Sódio)	2,71	1,95	0,19	31,44
RSC (meq/L)	–2,39	–1,00	–59,58	18,20
Dureza Total (T.H) (mg/L)	428,9	379,9	59,98	1879,8

Fonte: Autoria Própria

O valor médio do pH observado foi de 7,84, com variação entre 6,28 e 9,23, indicando uma tendência levemente alcalina. Parte das amostras está dentro dos limites aceitáveis para águas de irrigação, situando entre pH 6,5 e 8,5 (AYERS; WESTCOT, 1987). Valores mais elevados de pH, podem afetar a solubilidade de nutrientes no solo, comprometendo a absorção pelas plantas (CORDEIRO, 2000). A mediana dos dados foi similar à média, indicando que os valores extremos pouco afetaram a média como medida de tendência central.

A condutividade elétrica (C.E), parâmetro indicador da salinidade da água, apresentou média de 1.414,7 µS/cm, com valores máximos que atingiram 9.499 µS/cm. De acordo com a classificação proposta por Ayers e Westcot (1987), águas com condutividade superior a 1.000 µS/cm devem ser usadas com cautela, devido ao risco de acúmulo de sais no solo, o que pode comprometer o crescimento de culturas sensíveis. Esse dado é corroborado pelos altos níveis médios de sólidos totais dissolvidos (TDS), que alcançaram 905,4 mg/L, indicando salinidade significativa em grande parte das amostras. A diferença representativa entre a média aritmética

e a mediana de CE é reflexo da presença dos valores muito elevados, assim como ocorreu com o TDS.

No que tange à concentração de íons específicos, os dados mostraram teores elevados de cloreto (média de 205,6 mg/L), sódio (124,3 mg/L) e nitrato (80,5 mg/L), com máximos chegando a 2.480 mg/L, 748,1 mg/L e 1.028 mg/L, respectivamente. Valores de nitrato acima de 45 mg/L são considerados inaceitáveis para consumo humano segundo a Organização Mundial da Saúde (WHO, 2017), e sua presença em águas de irrigação em níveis elevados pode também contribuir para processos de eutrofização e toxicidade em culturas sensíveis. De modo geral, os elevados teores de íons cloreto, sódio e nitrato também influenciaram as médias aritméticas, tornando-as mais elevadas, comparadas às medianas. Este comportamento foi observado nas variáveis, a seguir, exceto para o RSC, em que a média aritmética foi menor que a mediana dos dados.

A razão de adsorção de sódio (RAS) variou entre 0,19 e 31,44, com média de 2,71. Esse índice é essencial para avaliar o risco de sodificação do solo, pois valores elevados indicam maior propensão à dispersão de partículas de argila, o que compromete a permeabilidade e a estrutura do solo (RICHARDS, 1954). A maior parte das amostras apresentou valores de RAS abaixo de 3, indicando baixo risco, mas há exceções críticas que merecem atenção no manejo.

O parâmetro RSC (resíduo de carbonatos), com média igual a –2,39 meq/L, apresentou valores predominantemente negativos, deslocando a média aritmética para valores menores. Esse resultado é desejado do ponto de vista da irrigação, pois valores negativos indicam baixa tendência à precipitação de carbonatos de cálcio e magnésio, favorecendo a disponibilidade desses nutrientes (AYERS; WESTCOT, 1987).

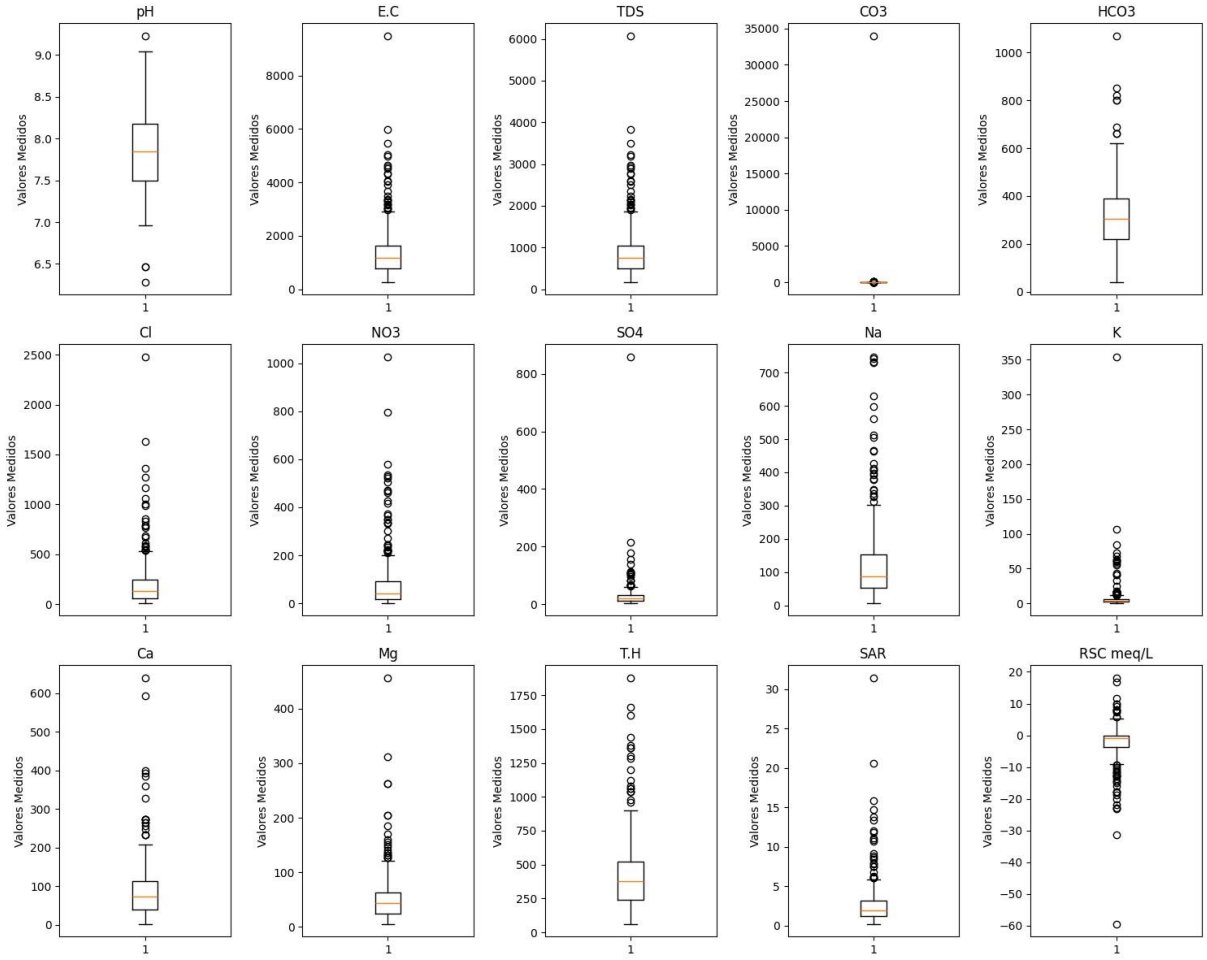
A dureza total da água, determinada pela concentração de íons cálcio e magnésio, apresentou valor médio elevado (428,9 mg/L como $CaCO_3$), caracterizando a água como "muito dura" segundo a classificação de Cavalcanti (2005). A presença de altos teores desses cátions pode ter implicações positivas, como a melhoria da estrutura do solo ao contrabalançar efeitos do sódio, mas também pode causar incrustações em sistemas de irrigação por gotejamento.

Os teores de íons flúor (F^-), embora apresentem média de 1,02 mg/L, ultrapassaram em alguns casos o limite de 1,5 mg/L recomendado para potabilidade pela OMS (WHO, 2017), sugerindo que essas amostras devem ser cuidadosamente avaliadas quanto ao uso em consumo humano e até mesmo para irrigação de plantas sensíveis.

Os boxplots das variáveis físico-químicas (Gráfico 1) revelaram a presença de diversos valores extremos em parâmetros como condutividade elétrica (E.C), sólidos dissolvidos totais (TDS), cloreto (Cl^-), potássio (K^+), RAS e RSC. Destacaram-se, por exemplo, um valor atípico

de carbonato (CO_3^{2-}) de 34,034 mg/L e valores de RSC altamente negativos, chegando a –59,58 meq/L, resultando em distribuições assimétricas e com caudas acentuadas. A maioria das variáveis apresentou assimetria à direita, como é o caso de E.C, TDS, Na^+ e SO_4^{2-}, enquanto o RSC exibiu cauda à esquerda. Essas características indicam variabilidade elevada entre as amostras. No contexto da classificação da qualidade da água utilizando algoritmos de aprendizado de máquina, a presença de *outliers* pode comprometer a acurácia, a precisão e o F1-score dos modelos, especialmente aqueles mais sensíveis a valores extremos, como as redes neurais artificiais (GERON, 2019). Por outro lado, modelos como árvores de decisão e florestas randômicas demonstram maior robustez a esse tipo de interferência, devido ao mecanismo de divisão binária e ao processo de votação entre múltiplas árvores (SIKDER; BATARSEH, 2021). Ao considerar a presente base de dados, por apresentar um número de amostras não muito elevado, decidiu-se mantê-la intacta para que seu treinamento e teste não fosse comprometido. Ou seja, buscou-se avaliar a influência da manutenção dos dados discrepantes após o processamento total dos dados. Dessa forma, organizou-se os dados em boxplots (Gráfico 1) para avaliação descritiva.

Gráfico 1: Boxplot dos dados das análises das águas de irrigação



Fonte: Autoria Própria

A análise conjunta dos boxplots apresentados no Gráfico 1 permite observar padrões importantes sobre a distribuição dos parâmetros físico-químicos avaliados. Nota-se que a maioria das variáveis apresenta distribuição assimétrica à direita, evidenciada por caudas superiores alongadas e uma concentração maior de valores abaixo da mediana. Parâmetros como condutividade elétrica (CE), sólidos totais dissolvidos (TDS), sódio (Na^+), cloreto (Cl^-) e nitrato (NO_3^-) exibem uma quantidade significativa de *outliers*, indicando que há amostras com concentrações excepcionalmente altas desses compostos. Isso pode sinalizar contaminações pontuais, uso inadequado de insumos agrícolas ou interferência antrópica em determinadas áreas de coleta (ALMEIDA, 2010; ALVIM; LIMA; SOUZA, 2020).

Essa variabilidade também sugere uma heterogeneidade no perfil químico das amostras, reforçando a necessidade de métodos robustos de modelagem, capazes de lidar com dados dispersos e complexos. Além disso, a diferença entre os quartis superiores e inferiores de variáveis como RAS e RSC aponta para comportamentos distintos entre os subgrupos de amostras — o que é indicativo de fontes de água com origens geológicas e ocupações antrópicas diferentes (AYERS; WESTCOT, 1987).

Por outro lado, variáveis como o pH e a dureza total apresentaram distribuição mais equilibrada, com menor presença de valores extremos, sugerindo uma maior estabilidade química nesses parâmetros entre as amostras. A presença de valores muito negativos de RSC (resíduo de carbonatos) — também destacados nos boxplots — indica uma baixa tendência à precipitação de carbonatos de cálcio e magnésio, o que é benéfico do ponto de vista agronômico, por favorecer a disponibilidade desses nutrientes no solo (AYERS; WESTCOT, 1987; RICHARDS, 1954).

De forma geral, os boxplots evidenciam o comportamento assimétrico e a ocorrência de valores discrepantes em variáveis críticas para a irrigação, o que justifica a aplicação de algoritmos de Aprendizado de Máquina, como árvores de decisão e florestas randômicas, que apresentam robustez frente a *outliers* e distribuições não normais (GERON, 2019; SIKDER; BATARSEH, 2021). Esses modelos se mostraram particularmente eficazes na presente pesquisa, conforme descrito nas etapas posteriores de modelagem e validação.

Devido ao banco de dados não ser extenso optamos por não remover os *outliers*, e assim não utilizar redes neurais artificiais que possuem alta sensibilidade a essa presença.

4.2. Aplicação do modelo de Árvore de Decisão

Para estruturar o modelo de Aprendizado de Máquina, a base de dados foi dividida em dois subconjuntos: um para treinamento (75% das amostras) e outro para teste (25%). Essa

divisão foi feita de forma aleatória, garantindo a representatividade dos dados em ambas as partes.

O percentual de 25% foi utilizado para compor o conjunto de teste, prática comum em estudos com aprendizado de máquina, pois permite avaliar a capacidade de generalização do modelo. Para manter a consistência na divisão dos dados durante os testes, foi adotado um critério de aleatoriedade fixo.

A otimização dos hiperparâmetros do modelo foi realizada por meio da técnica de validação cruzada com busca em grade (grid search), utilizando somente os dados de treinamento. Esse procedimento permite testar diferentes combinações de parâmetros e selecionar aquela com melhor desempenho médio, sem recorrer ao conjunto de teste.

Foram testados diferentes hiperparâmetros, como o critério para divisão dos nós (“gini” e “entropia”), profundidade máxima da árvore, número mínimo de amostras por divisão e por nó folha. Embora o critério “entropia” seja baseado em conceitos de teoria da informação, ele não foi detalhado neste trabalho, pois o foco esteve na aplicação prática dos modelos.

A utilização do GridSearchCV resultou na seleção de um conjunto de hiperparâmetros que maximizou o desempenho do modelo, evidenciando a importância do ajuste fino para a melhoria da capacidade de classificação. Esses parâmetros, definidos após a validação cruzada, foram então utilizados para treinar o modelo final e avaliar sua performance no conjunto de teste.

A escolha dos parâmetros influencia diretamente no resultado do treinamento, para isso utilizamos a biblioteca Scikit-learn para analisar e selecionar os melhores parâmetros para obtenção da melhor acurácia. Os parâmetros utilizados foram: critério (Gini), profundidade (2), indicando a influência de poucas variáveis em sua estrutura, número mínimo de amostras para dividir um nó interno (2) e número mínimo de amostras para os nós folha (1).

Além da obtenção dos melhores parâmetros realizamos a validação cruzada entre as combinações de parâmetros e obtivemos uma acurácia de validação igual a 0,989.

Com os melhores hiperparâmetros definidos, foi implementado um processo de avaliação baseado em 100 repetições de treinamento e teste com diferentes divisões aleatórias dos dados. Esse procedimento visou avaliar a estabilidade dos resultados e reduzir a influência de variações ocasionais na amostragem.

Em cada execução do *loop*, foram realizados os seguintes procedimentos:

- Primeiramente, as bases de treinamento e teste foram redefinidas por meio da função train_test_split(), da biblioteca Scikit-learn, adotando-se uma proporção de 75% para treino e 25% para teste (test_size=0.25), garantindo diferentes amostragens a cada repetição;
- Em seguida, o modelo classificador, já configurado com os hiperparâmetros otimizados, foi instanciado como um objeto, sendo treinado com o método fit(), utilizando os dados de treinamento;
- O processo de predição foi executado sobre o conjunto de teste por meio do método predict(), gerando a base de valores previstos (c_pred), que posteriormente foi comparada com os rótulos reais (c_test).

A referida metodologia foi fundamental na implementação do modelo de árvore de decisão, pois garantiu que a média aritmética de 100 processamentos, gerando bases de treinamento e de teste aleatórias, representasse a eficácia. Isso evitou que um único processamento aleatório fosse responsável pela seleção ou destarte do método.

A acurácia de treinamento foi calculada utilizando o método score(), enquanto a acurácia de previsão foi obtida por meio da função accuracy_score(), ambas da biblioteca Scikit-learn. Para as demais métricas de avaliação (precisão, revocação e F1-score), foram gerados vetores vazios que, por meio da função append() foram preenchidos com os respectivos resultados a cada iteração entre as 100 estabelecidas.

Ao final das 100 execuções, os valores médios e os desvios-padrão de cada métrica foram calculados utilizando os métodos mean() e std(), ambos pertencentes à biblioteca NumPy. Para facilitar a apresentação e a interpretação dos resultados, os valores finais foram arredondados para três casas decimais com o uso do método round(), como ilustrado na Tabela 3.

Tabela 3: Dados descritivos das métricas do modelo de Árvore de Decisão considerando 100 processamentos

Acurácia de treinamento:	0.995 +/- 0.002
Acurácia de previsão:	0.99 +/- 0.009
Precisão por categoria:	
C2:	1.0 +/- 0.0
C3:	0.997 +/- 0.006
C4:	0.939 +/- 0.061
O.G:	0.0 +/- 0.0
Recall por categoria	
C2:	1.0 +/- 0.0
C3:	0.996 +/- 0.007
C4:	0.986 +/- 0.031
O.G:	0.0 +/- 0.0
F1-score por categoria:	
C2:	1.0 +/- 0.0
C3:	0.997 +/- 0.004
C4:	0.962 +/- 0.034
O.G:	0.0 +/- 0.0

Fonte: Autoria Própria

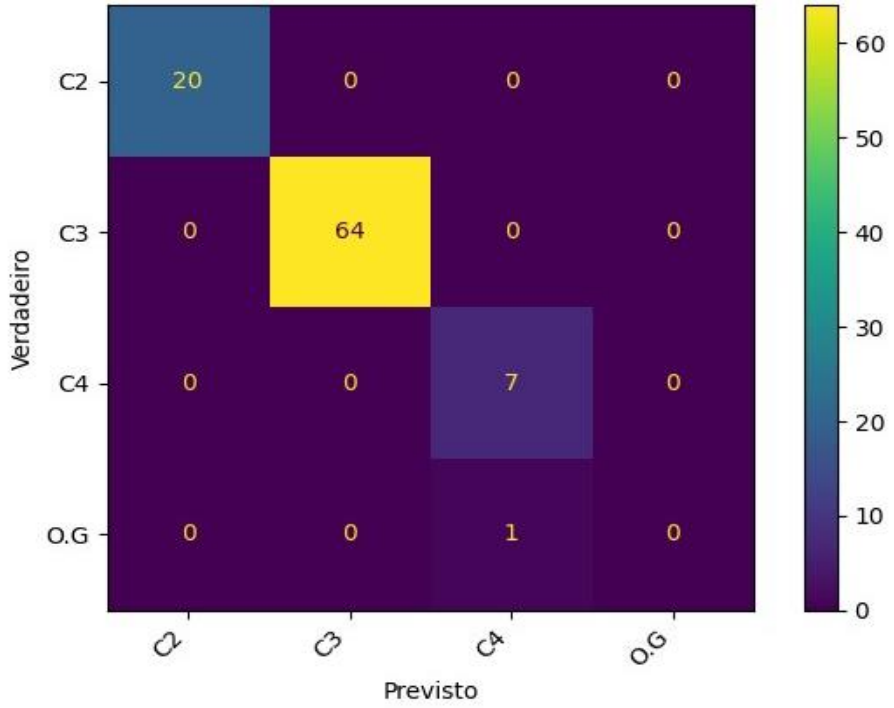
A base de dados de treinamento apresentou elevada acurácia acertando grande parte das classificações. A acurácia de teste também foi alta, destacando-se que 25% dos registros foram utilizados, um volume representativo de dados. Assim, se o atual modelo fosse utilizado na avaliação de novas amostras de água, espera-se sucesso nas classificações corretas próximo a 99%.

A grande eficácia de classificação observada, de modo geral, na precisão e no recall gerou ótimos resultados para F1-score. Por se tratar de uma média harmônica entre a precisão e o recall, valores mais elevados de F1-score indicam que o modelo apresenta um bom equilíbrio entre esses dois aspectos, ou seja, apresenta baixos valores de classificações falsas (FP e FN).

De forma complementar e com o objetivo de proporcionar uma melhor visualização dos resultados discutidos, apresenta-se no Gráfico 2 a matriz de confusão obtida em uma das execuções do modelo de Árvore de Decisão, utilizando os dados de teste. Essa matriz ilustra,

de maneira didática, como o modelo realizou a classificação das amostras nas diferentes categorias.

Gráfico 2: Matriz de confusão para o modelo de Árvore de Decisão



Fonte: Autoria Própria

Observa-se que o número de classificações verdadeiras positivas para a Classe C2 e C3 é bastante representativo em relação ao total de registros presentes no conjunto de teste. Esse desempenho favorável contribui diretamente para os valores elevados das métricas de precisão e F1-score associados a essa classe (Tabela 3), refletindo a capacidade do modelo em identificar corretamente as amostras pertencentes a essa categoria. Como destacado na discussão dos hiperparâmetros otimizados, a profundidade da Árvore de Decisão foi baixa, indicando que poucas variáveis foram determinantes para a efetiva classificação das amostras.

A classe C4 (Tabela 3), por sua vez, apresenta um pouco menos de eficácia nas métricas indicando que quando os valores de condutividades são muito elevados o modelo de Árvore de Decisão tem mais dificuldade de acerto. A classe O.G, para amostras que não se encaixam no sistema de classificação, além de atípicas, foram relacionadas a poucas amostras na base de dados, motivos pelos quais o algoritmo não conseguiu classificá-la corretamente.

Por fim, destaca-se que as variáveis estudadas, sem considerar a condutividade elétrica, levou o modelo de Árvore de Decisão a utilizar prioritariamente as informações de sólidos totais dissolvidos para realizar a classificação dos dados. Este resultado é coerente com o fato de que parte considerável dos sólidos totais dissolvidos em águas de irrigação são é formada por compostos iônicos, relacionados diretamente com a condutividade eletrônica.

4.3. Aplicação do modelo de Floresta Randômica

Após a análise com o modelo de Árvore de Decisão, foi aplicado o modelo de Floresta Randômica com o objetivo de avaliar se a combinação de múltiplas árvores melhoraria a robustez e a acurácia da classificação da qualidade das águas. Esse modelo é conhecido por sua capacidade de lidar com dados de alta dimensionalidade e por ser resistente a sobreajuste (BREIMAN, 2001).

Assim como no modelo anterior, os dados foram divididos em conjuntos de treinamento e teste, adotando-se 75% das amostras para o treinamento e 25% para o teste. Essa divisão foi repetida diversas vezes a fim de mitigar os efeitos da aleatoriedade sobre os resultados (GÉRON, 2019).

Para garantir um modelo confiável, foi realizada a otimização dos hiperparâmetros do algoritmo utilizando validação cruzada com busca em grade (*Grid Search*). Os parâmetros avaliados incluíram:

- Número de árvores no modelo (*n_estimators*);
- Profundidade máxima das árvores (*max_depth*);
- Número de atributos considerados em cada divisão (*max_features*);
- Número mínimo de amostras por folha (*min_samples_leaf*).

Essa otimização foi conduzida utilizando apenas os dados de treinamento, conforme boas práticas recomendadas para evitar vazamento de dados (MÜLLER; GUIDO, 2016). A métrica utilizada para avaliação durante a busca foi a acurácia média obtida nas validações internas.

Após a seleção dos melhores hiperparâmetros, o modelo final foi testado quanto à sua estabilidade. Para isso, foi realizado um processo de 100 execuções independentes, no qual:

1. Os dados foram novamente particionados aleatoriamente em treinamento e teste;
2. O modelo foi treinado com os dados de treinamento e os hiperparâmetros otimizados;
3. O modelo foi testado com o conjunto de teste, e foram calculadas as métricas: acurácia, precisão, revocação e F1-score;
4. Os valores obtidos em cada execução foram armazenados para análise estatística posterior.

As métricas foram calculadas de acordo com as definições clássicas da área (MURPHY, 2012), e os resultados foram expressos como média ± desvio-padrão. Essa abordagem evita que

os resultados sejam influenciados por uma única divisão aleatória dos dados, conforme sugerido por Géron (2019) e Choudhary (2021).

A Tabela 4 apresenta os valores médios e desvios-padrão das métricas de avaliação obtidas.

Tabela 4 – Resultados médios e desvios padrão do modelo de Floresta Randômica com 100 execuções

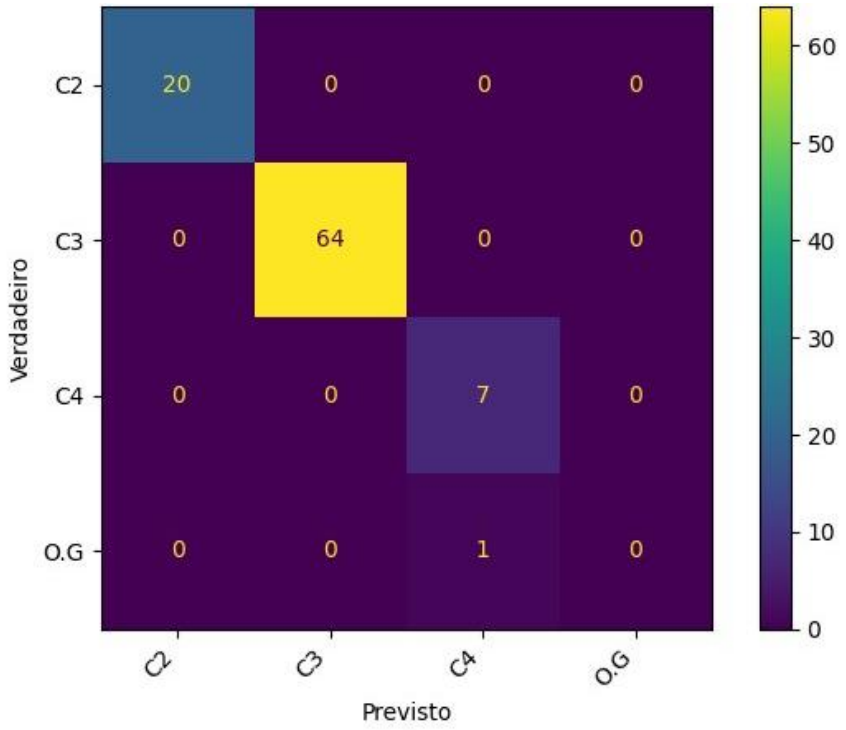
Métrica	**Média**	**Desvio-padrão**
Acurácia de treinamento	0,999	±0,001
Acurácia de previsão	0,993	±0,007
Precisão – C2	1,000	±0,000
Precisão – C3	0,998	±0,004
Precisão – C4	0,957	±0,049
Precisão – O.G	0,000	±0,000
F1-score – C2	1,000	±0,000
F1-score – C3	0,998	±0,003
F1-score – C4	0,970	±0,027
F1-score – O.G	0,000	±0,000

Fonte: Autoria própria

O desempenho do modelo de Floresta Randômica também foi extremamente elevado, com destaque para as Classes C2 e C3, que apresentaram métricas praticamente perfeitas. A acurácia de previsão se manteve próxima de 99%, com desvio-padrão reduzido, evidenciando a estabilidade do modelo ao longo das 100 execuções. A Classe C4, embora com bons resultados, apresentou maior variabilidade nas métricas, o que pode estar associado à menor representatividade dessa classe no conjunto de dados. Já a classe O.G, por apresentar muito poucas amostras, não teve êxito em sua classificação.

A matriz de confusão apresentada no Gráfico 3 ilustra os acertos e erros de classificação em uma das execuções do modelo, confirmando a predominância dos verdadeiros positivos nas classes principais e a ausência de previsões corretas para a classe O.G.

Gráfico 3 – Matriz de confusão para floresta randômica



Fonte: Autoria própria

Os resultados obtidos reforçam que o modelo de Floresta Randômica apresenta desempenho elevado e estável, mesmo com a presença de outliers e com dados não extensos. Essa característica também indica o potencial da Floresta Randômica na classificação da qualidade de águas de irrigação, conforme evidenciado em estudos prévios (SIKDER; BATARSEH, 2021).

Durante a análise exploratória dos dados, identificou-se a presença de diversos *outliers*, sobretudo nas variáveis de sólidos dissolvidos totais, bem como nos teores de íons sódio e cloreto. Apesar disso, o modelo de Floresta Randômica apresentou desempenho excepcional, com média das métricas maior que 95,0 % para as classes C2, C3 e C4. Tal desempenho evidencia a robustez intrínseca do método *ensemble*, o qual, ao construir múltiplas árvores com subconjuntos aleatórios de amostras e atributos, minimizando o impacto de valores extremos.

A literatura corrobora esses achados: Fang Xin et al. (2021) utilizaram Floresta Randômica para modelar os efeitos de poluentes em uma lagoa tropical, com dados ruidosos e assimétricos, alcançando acurácia de 99,95%. De forma semelhante, Choudhury et al. (2023) reportaram 98,99% de acurácia na classificação de águas subterrâneas na Índia, mesmo diante de *outliers* naturais em variáveis como ferro e nitrato. Por outro lado, Han et al. (2021) observaram queda no desempenho de Árvores de Decisão quando expostas a valores extremos em parâmetros como turbidez e coliformes. Complementarmente, Shrestha et al. (2021)

evidenciaram a menor capacidade de generalização dos modelos de árvore única frente a distribuições assimétricas.

Para definir o melhor modelo de classificação, foi utilizado o teste de Wilcoxson (α = 0,05) para comparação das medianas dos dados (Tabela 5).

Tabela 5 – Comparação entre os modelos de classificação

Modelo	Mediana	p-valor	Resultado
Acurácia de Previsão			
Árvore de Decisão	0,989	0,503	Estatisticamente iguais
Floresta Randômica	0,989		
Previsão			
AD C2	0,999	0,617	Estatisticamente iguais
FR C2	0,999		
Recall			
AD C3	0,999	0,617	Estatisticamente iguais
FR C3	0,999		
F1-Score			
AD C4	0,999	0,617	Estatisticamente iguais
FR C4	0,999		

Sendo: AD = Árvore de Decisão; FR = Floresta Randômica; C2, C3 e C4 = classes de águas de irrigação de acordo com a condutividade elétrica.

Fonte: Autoria Própria

Os resultados da Tabela 5 indicam que ambos os modelos apresentam a mesma eficácia com relação às métricas estudadas, com 95% de confiabilidade. Destaca-se que este comportamento se deve possivelmente à grande importância da variável sólidos totais dissolvidos para a classificação dos dados, dominando a Árvore de Decisão e refletindo sua importância na Floresta Randômica.

Assim sendo, ambos os modelos podem ser usados na classificação de águas de irrigação, tomando como referência a base de dados utilizada no presente estudo.

5. CONCLUSÃO

O presente trabalho apresentou uma análise detalhada do processo de desenvolvimento de um modelo de Aprendizado de Máquina, aplicado a análises de águas que possuem uma quantia vasta de parâmetros a serem mensurados. Os resultados obtidos foram satisfatórios, evidenciando que, com uma preparação adequada, o modelo foi capaz de captar padrões complexos, alcançando ótimos índices de acurácia, tanto no treinamento quanto na validação.

A análise demonstrou que mesmo não havendo a eliminação de *outliers* com base no critério do boxplot não houve um comprometimento negativo da acurácia do modelo.

Observou-se elevadas acurácias, bem como precisão, recall e f1-score relacionados às classes C2, C3 e C4 para ambos os modelos testados. Ao comparar as métricas por meio do teste de Wilcoxon ($\alpha = 0,05$), observou-se que ambos os modelos possuem a mesma eficiência na classificação dos dados, indicando a Árvore de Decisão como mais adequado por ser menos complexo.

Como perspectivas futuras, recomenda-se a construção de um banco de dados com uma quantidade vasta de amostras e de diversas regiões do Brasil, pois, como a composição da água está relacionada à composição do solo, quanto maior a distância entre os locais, maiores diferenças de composição presentes.

REFERÊNCIAS

ALMEIDA, O. A. de. **Qualidade da água de irrigação**. Cruz das Almas: Embrapa Mandioca e Fruticultura, 2010. Disponível em: http://www.cnpmf.embrapa.br/publicacoes/livro_qualidade_agua.pdf. Acesso em: 21 jan. 2011.

ALVIM, D. S., LIMA, A. P., & SOUZA, T. M. **Análise da qualidade da água e sua correlação com parâmetros físico-químicos.** Revista Brasileira de Recursos Hídricos, v. 25, n. 4, p. 1-15, 2020. Disponível em: https://www.abrh.org.br. Acesso em: 29 out. 2024.

AYERS, R. S.; WESTCOT, D. W. **La calidad del agua para agricultura**. Roma: FAO, 1987. 174 p. (Estudos FAO: Riegos y Drenajes, 29).

BELIZÁRIO, Talita Lucas; SOARES, Mara Alves; ASSUNÇÃO, Washington Luiz. **Qualidade da água para irrigação no projeto de assentamento Dom José Mauro, Uberlândia-MG**. Revista Getec, v. 3, n. 5, p. 53-73, 2014. Disponível em: https://revistas.fucamp.edu.br/index.php/getec/article/view/430/344. Acesso em: 14 fev. 2025.

BIAU, Gérard; SCORNET, Erwan. A Random Forest Guided Tour. arXiv, 18 nov. 2015. Disponível em: https://arxiv.org/abs/1511.05741. Acesso em: 20 jun. 2025.

BISHOP, C. M. **Pattern Recognition and Machine** Learning. New York: Springer, 2006. Disponível em: https://www.springer.com/gp/book/9780387310732. Acesso em: 08 fev. 2025.

BOSTROM, N**. Superintelligence: Paths, Dangers, Strategies.** Oxford: Oxford University Press, 2014.

BREIMAN, L. **Random Forests**. Machine Learning, v. 45, n. 1, p. 5-32, 2001. Disponível em: https://www.stat.berkeley.edu/~breiman/RandomForests/. Acesso em: 04 fev. 2025.

BRYNJOLFSSON, Erik; MCAFEE, Andrew. **Machine, Platform, Crowd: Harnessing Our Digital Future**. New York: W.W. Norton & Company, 2017.

BUI, D. T., NAGARAJAN, R., TRAN, Q. A., KUMAR, A., HO, L. S., & LEE, S. **A novel artificial intelligence approach based on neural fuzzy inference model for flood susceptibility modeling.** Journal of Hydrology, v. 581, p. 124388, 2020. Disponível em: https://www.sciencedirect.com. Acesso em: 29 out. 2024.

CHAOUDHARY, Divya. Bootstrapping and OOB samples in Random Forests. Analytics Vidhya, 18 abr. 2021. Disponível em: https://medium.com/analytics-vidhya/bootstrapping-andoob-samples-in-random-forests-6e083b6bc341. Acesso em: 20 jun. 2025.

CHOLLET, François. **Deep Learning with Python**. 2. ed. Shelter Island: Manning Publications, 2018.

CONSELHO NACIONAL DO MEIO AMBIENTE (Brasil). Resolução nº 357, de 17 de março de 2005. **Dispõe sobre a classificação dos corpos de água e diretrizes ambientais para seu enquadramento, bem como estabelece as condições e padrões de lançamento de efluentes.**

CORDEIRO, G. G. **Qualidade da água para fins de irrigação**. Petrolina: Embrapa Semiárido, 2000. (Circular Técnica, 36). Disponível em: https://www.infoteca.cnptia.embrapa.br/bitstream/doc/153785/1/CircularTecnica36.pdf. Acesso em: 17 jun. 2025.

CORTEZ, Paulo; CERDEIRA, António; ALMEIDA, Fernando; MATOS, Telmo; REIS, José. **Modeling wine preferences by data mining from physicochemical properties**. Decision Support Systems, v. 47, p. 547-533, 2009. DOI: 10.1016/j.dss.2009.05.016.

CURTIN, J. Justin. Introduction to Applied Machine Learning - Random Forest. 2025. Disponível em: https://jjcurtin.github.io/book_iaml/l09_random_forest.html. Acesso em: 20 jun. 2025.

FANG, Xin; ZHANG, Yifei; et al. **Random forest-based understanding and predicting of the impacts of anthropogenic nutrient inputs on the water quality of a tropical lagoon.** Environmental Research Letters, v. 16, n. 5, art. 055003, 16 abr. 2021. DOI: 10.1088/1748-9326/abf395. Disponível em: https://doi.org/10.1088/1748-9326/abf395. Acesso em: 29 jun. 2025.

FLORIDI, Luciano et al. **AI4People – An Ethical Framework for a Good AI Society: Opportunities, Risks, Principles, and Recommendations. Minds and Machines**, v. 28, n. 4, p. 689–707, 2018.

GAO, Alice. **Lecture 7 – Decision Trees**. University of Toronto, 2021. Disponível em: https://www.cs.toronto.edu/~axgao/cs486686_f21/lecture_notes/Lecture_07_on_Decision_Trees.pdf. Acesso em: 20 jun. 2025.

GARCIA, Cleverson Fernando**. Avaliação de diferentes modelos de Machine Learning para a classificação da qualidade de vinhos.** 2022. Monografia (Especialização em Ciência de Dados e Big Data) – Pontifícia Universidade Católica de Minas Gerais, Belo Horizonte, 2022.

GARLADINNE, Sivapriya. **Telangana Post Monsoon Ground Water Quality Data.** Kaggle, [s.d.]. Disponível em: https://www.kaggle.com/datasets/sivapriyagarladinne/telangana-postmonsoon-ground-water-quality-data. Acesso em: 11 maio 2025.

GÉRON, Aurélien. **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems**. 2. ed. Sebastopol: O'Reilly Media, 2019.

GERVÁSIO, E. S.; CARVALHO, J. A.; SANTANA, M. J. **Efeito da salinidade da água de irrigação na produção da alface americana.** Revista Brasileira de Engenharia Agrícola e Ambiental. 2021. Disponível em: https://www.scielo.br/j/rbeaa/a/LrRyFtMDYwhZKTPk48Spjpn/. Acesso em: 21 jun. 2025.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. **Deep Learning.** Cambridge: MIT Press, 2016.

Groundwater Quality Assessment Process. Water Resources Management, v. 38, n. 2, p. 621–637, 2024. DOI: 10.1007/s11269-023-03690-y. Disponível em: https://link.springer.com/article/10.1007/s11269-023-03690-y. Acesso em: 29 jun. 2025.

HAN, Jing; et al. **Effect of *outliers* on decision tree performance in water quality classification.** Water Research, v. 198, art. 117154, 2021. DOI: 10.1016/j.watres.2021.117154. Disponível em: https://doi.org/10.1016/j.watres.2021.117154. Acesso em: 29 jun. 2025.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. **The Elements of Statistical Learning**: Data Mining, Inference, and Prediction. 2. ed. New York: Springer, 2009. Disponível em: https://web.stanford.edu/~hastie/ElemStatLearn/. Acesso em: 04 fev. 2025.

https://iwaponline.com/ws/article/21/8/4015/82364/Soft-computing-technique-based-prediction-of-water. Acesso em: 21 jun. 2025.

https://www.mdpi.com. Acesso em: 29 out. 2024.

ITABORAHY, C. R. et al. **Agricultura irrigada e o uso racional da água.** Brasília: Agência Nacional das Águas, Superintendência de Conservação de Água e Solo, 2004.

IUNES, João Paulo. **Árvore de decisão**. Colaborae, 19 jul. 2023. Disponível em: https://colaborae.com.br/blog/2023/07/19/arvore-de-decisao. Acesso em: 24 jul. 2025.

JAIN, A. K.; MURTY, M. N.; FLYNN, P. J. **Data Clustering**: A Review. ACM Computing Surveys, v. 31, n. 3, p. 264–323, 1999. Disponível em: https://dl.acm.org/doi/10.1145/331499.331504. Acesso em: 08 fev. 2025.

JAMES, G.; WITTEN, D.; HASTIE, T.; TIBSHIRANI, R. **An Introduction to Statistical Learning:** with Applications in R. New York: Springer, 2013. Disponível em: https://www.statlearning.com/. Acesso em: 04 fev. 2025.

JORDAN, M. I.; MITCHELL, T. M. **Machine learning: Trends, perspectives, and prospects.** Science, v. 349, n. 6245, p. 255–260, 2015. DOI: 10.1126/science.aaa8415. Disponível em: https://www.science.org/doi/10.1126/science.aaa8415. Acesso em: 21 jun. 2025.

KAPLAN, Andreas; HAENLEIN, Michael. Siri, **Siri, in my Hand: Who's the Fairest in the Land? On the Interpretability of Artificial Intelligence**. Business Horizons, v. 62, n. 1, p. 15–25, 2019.

LECUN, Y.; BENGIO, Y.; HINTON, G. **Deep learning**. Nature, v. 521, p. 436–444, 2015. DOI: 10.1038/nature14539. Disponível em: https://www.nature.com/articles/nature14539. Acesso em: 21 jun. 2025.

LI, Y., SUN, Z., & WANG, J. **Water quality prediction based on machine learning algorithms**: a case study in China. Water, v. 14, n. 3, p. 310-320, 2022. Disponível em:
LOUPPE, Gilles. Understanding Random Forests: From Theory to Practice. arXiv, 28 jul. 2014. Disponível em: https://arxiv.org/abs/1407.7502. Acesso em: 20 jun. 2025.

MAGALHÃES, Marcele Souza; PINHEIRO, Antonio Anderson Freitas; RODRIGUES, Frederico de Medeiros; MESQUITA, Paulo Roberto Ribeiro. **Análise da qualidade da água utilizada para irrigação em propriedades rurais de Cravolândia, BA, Brasil**. Research, Society and Development, v. 11, n. 11, e314111133521, 2022. DOI: http://dx.doi.org/10.33448/rsd-v11i11.33521. Acesso em: 14 fev. 2025.

MENDES FILHO, H. T. et al. **Uso de águas de baixa qualidade para irrigação no Nordeste brasileiro: uma revisão**. Water Resources and Irrigation Management, v. 14, n. 1-3, 2024.

DOI: 10.19149/wrim.v14i1-3.5258. Disponível em: https://doi.org/10.19149/wrim.v14i13.5258. Acesso em: 21 jun. 2025.

MITCHELL, Tom M. **Machine Learning**. New York: McGraw Hill, 1997.

MMA. Ministério do Meio Ambiente e Mudança do Clima. Disponível em: https://www.mma.gov.br. Acesso em: 22 jan. 2025.

MOOREN, Sander. Living Textbook – Random Forest. University of Twente, 2025. Disponível em: https://ltb.itc.utwente.nl/798/concept/154676. Acesso em: 20 jun. 2025.

MÜLLER, Andreas C.; GUIDO, Sarah. **Introduction to Machine Learning with Python: A Guide for Data Scientists**. Sebastopol: O'Reilly Media, 2016.

MURPHY, Kevin P. **Machine Learning: A Probabilistic Perspective.** Cambridge: MIT Press, 2012.

NIELSEN, M. **Neural Networks and Deep Learning**. Detemination Press, 2015 Disponível em: http://neuralnetworksanddeeplearning.com/. Acesso em: 04 fev. 2025.

NUR, Fatma. **Introduction to Machine Learning. Medium**, 12 mar. 2024. Disponível em: https://medium.com/@fatma.nur/introduction-to-machine-learning-c829d5c3803e. Acesso em: 11 maio 2025.

ONU. **Transformando Nosso Mundo**: A Agenda 2030 para o Desenvolvimento Sustentável. Organização das Nações Unidas, 2015. Disponível em: https://www.un.org. Acesso em: 29 out. 2024.

PANG, J., WANG, D., & WU, Q. **Classification of water quality using machine learning techniques**. Environmental Monitoring and Assessment, v. 193, n. 10, p. 1-12, 2021. Disponível em: https://link.springer.com. Acesso em: 29 out. 2024.

PEREIRA DE SOUSA, T. et al. **Análise da qualidade da água de irrigação em função de sua condutividade elétrica**. Agropecuária Científica no Semi-Árido, v. 10, n. 3, 2021. DOI: 10.30969/acsa.v10i3.568. Disponível em: https://acsa.revistas.ufcg.edu.br/acsa/index.php/ACSA/article/view/568. Acesso em: 21 jun. 2025.

PIZARRO, F. **Riegos localizados de alta frecuencia (RLAF): goteo, microaspersión y exudación.** 3. ed. Madrid: Mundi-Prensa, 1996.

RICHARDS, L. A. (Ed**.). Diagnosis and improvement of saline and alkali soils.** Washington: United States Department of Agriculture, 1954. (USDA Agriculture Handbook, 60). Disponível em: https://www.ars.usda.gov/ARSUserFiles/20360500/hb60_pdf/hb60complete.pdf. Acesso em: 17 jun. 2025.

RUBIO, J. J. et al. **Sistema sensor para el monitoreo ambiental basado en redes neuronales**. arXiv, 28 abr. 2019. Disponível em: https://arxiv.org/abs/1904.12234. Acesso em: 20 jun. 2025.

RUSSELL, Stuart J.; NORVIG, Peter. **Artificial Intelligence: A Modern Approach**. 4. ed. Boston: Pearson, 2021.

SAMUEL, Arthur L. **Some Studies in Machine Learning Using the Game of Checkers.** IBM Journal of Research and Development, v. 3, n. 3, p. 210–229, 1959.

SANTOS, J. R. **A salinidade na agricultura irrigada: teoria e prática**. Campina Grande: UFPB, 2000.

SEGATTO, Vinicius. **Floresta Aleatória.** Medium, 10 out. 2023. Disponível em: https://medium.com/@viniciussegatto11/floresta-aleatória-01f3fe9c5c1b. Acesso em: 24 jul. 2025.

SHALEV-SHOWTZ, S.; BEN-DAVID, S. **Understanding Machine Learning**: From Theory to Algorithms. Cambridge: Cambridge University Press, 2014. Disponível em: http://www.cs.huji.ac.il/~shais/UnderstandingMachineLearning/. Acesso em: 04 fev. 2025.

SHRESTHA, R.; et al. **Performance evaluation of single and ensemble tree models in water quality assessment under asymmetric data**. Environmental Monitoring and Assessment, v. 193, n. 3, art. 155, 2021. DOI: 10.1007/s10661-021-08901-z. Disponível em: https://doi.org/10.1007/s10661-021-08901-z. Acesso em: 29 jun. 2025.

SIKDER, M. N. K.; BATARSEH, F. A**. Outlier Detection using AI: a survey. *arXiv,*** 1 dez. 2021. Disponível em: https://arxiv.org/abs/2112.00588. Acesso em: 19 jun. 2025.

SILVA, Í. N. et al. **Qualidade de água na irrigação**. Agropecuária Científica no Semi-Árido, v. 7, n. 3, p. 134–144, 2019. DOI: 10.30969/acsa.v7i3.134. Disponível em: https://doi.org/10.30969/acsa.v7i3.134. Acesso em: 21 jun. 2025.

SIMÃO, M. S.; PÉCORA JUNIOR, J. E. **Inteligência Artificial aplicada ao tratamento de efluentes: uma revisão da produção científica internacional nos últimos cinco anos.**

Organizações & Sustentabilidade, v. 8, n. 2, p. 68–77, jul./dez. 2020. DOI: 10.5433/2318-9223.2020v8n2p68. Disponível em: https://ojs.uel.br/revistas/uel/index.php/ros/article/view/40690. Acesso em: 20 jun. 2025.

SINGH, B.; SIHAG, P.; SINGH, V. P.; SEPAHVAND, A.; SINGH, K.; et al. **Soft computing techniques-based prediction of water quality index.** Water Supply, v. 21, n. 8, p. 4015–4029, dez. 2021. DOI: 10.2166/ws.2021.157. Disponível em:

SUTTON, Richard S.; BARTO, Andrew G. **Reinforcement Learning: An Introduction.** 2. ed. Cambridge: MIT Press, 2018.

TensorFlow: conceitos, ferramentas e técnicas para a construção de sistemas inteligentes. 2. ed. Rio de Janeiro: Alta Books, 2019.

TOPOL, Eric. **Deep Medicine: How Artificial Intelligence Can Make Healthcare Human Again.** New York: Basic Books, 2019.

TURING, Alan M. **Computing Machinery and Intelligence**. Mind, v. 59, n. 236, p. 433–460, 1950

WHO. **Water Quality and Health – Review of Current Practices and Recommendations for Future Monitoring**. World Health Organization, 2022. Disponível em: https://www.who.int. Acesso em: 29 out. 2024.

WORLD HEALTH ORGANIZATION (WHO). **Guidelines for drinking-water quality.** 4th ed. Geneva: WHO, 2017. Disponível em: https://www.who.int/publications/i/item/9789241549950. Acesso em: 17 jun. 2025.

ZEGGAR, Aymen; OUNOKI, Samira; TELLI, Abdelmoutia; et al. **Machine Learning For Groundwater Quality Classification: A Step Towards Economic and Sustainable**

ANEXO: Script em Python

```
[ ]  import pandas as pd
     dados = pd.read_csv("Planilha 1.csv", sep = ";", header=0)
     dados_cl = pd.read_csv("Planilha_text.csv", sep = ";", header=0)
     print(dados)
     print(dados_cl)
```

```
[ ]  print('Dimensões da base de dados: ', dados.shape)
     print('Dimensões da base de dados: ', dados_cl.shape)
```

```
[ ]  #Verificação de dados faltantes

     print(dados.isnull(), '\n', '\n') # Mostra todos

     print(dados.isnull().sum()) # Mostra a soma dos dados faltantes
```

```
[ ]  #Verificação de outros tipos de dados

     dados.select_dtypes(exclude='number')
```

```
[ ]  # Tabela com os dados descritivos dos atributos preditores

     dados.describe()
```

```
#Padronização dos dados do dataframe

from sklearn.preprocessing import StandardScaler
import pandas as pd # Import pandas just in case it wasn't imported earlier in the session
import numpy as np # Import numpy

# Substitua vírgulas por pontos em todas as colunas e converta para numérico
for col in dados.columns:
    # Check if the column is of object type before attempting string operations
    if dados[col].dtype == 'object':
        dados[col] = dados[col].astype(str).str.replace(',', '.', regex=False) # Use regex=False for literal replacement
        dados[col] = pd.to_numeric(dados[col], errors='coerce') # Convert to numeric, coercing errors to NaN

# Now, proceed with standardization only on numeric columns to avoid errors
# Select only numeric columns for standardization
numeric_cols = dados.select_dtypes(include=np.number).columns
dados_numeric = dados[numeric_cols]

pad = StandardScaler()
dados_p = pad.fit_transform(dados_numeric) # Fit and transform only numeric data
print(dados_p)
```

```
[ ]  dados_p_df = pd.DataFrame(dados_p, columns=['pH','E.C', 'TDS', 'CO3',
                'HCO3', 'Cl', 'NO3' , 'SO4' , 'Na' , 'K' , 'Ca' , 'Mg' ,
                'TH' , 'SAR' , 'RSC (meq/L)'])
     dados_p_df
```

```python
# Criar box plots para todas as colunas do dataframe
import matplotlib.pyplot as plt

num_cols = dados.shape[1]
cols_per_row = 5
num_rows = (num_cols + cols_per_row - 1) // cols_per_row

fig, axes = plt.subplots(num_rows, cols_per_row, figsize=(15, num_rows * 4))
axes = axes.flatten()

for i, col in enumerate(dados.columns):
    axes[i].boxplot(dados[col])
    axes[i].set_title(col)
    axes[i].set_ylabel('Valores Medidos')

# Ocultar subplots vazios, se houver
for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])

plt.tight_layout()
plt.show()
```

```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, accuracy_score, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

# Separando X (atributos preditores) e y (classe)
X_dados = dados.iloc[:, 0:15].values  # Colunas de pH até RSC
y_dados = dados_cl['Classification 2'].values  # Coluna de classe

print('Dimensão dos atributos preditores:', X_dados.shape)
print('Dimensão do atributo classe:', y_dados.shape)

# Padronização dos dados
pad = StandardScaler()
X_dados_p = pad.fit_transform(X_dados)

# Divisão em treino e teste
X_train, X_test, y_train, y_test = train_test_split(X_dados_p, y_dados, test_size=0.25, random_state=42)

print('\nDimensão X_train:', X_train.shape)
print('Dimensão X_test:', X_test.shape)
print('Dimensão y_train:', y_train.shape)
print('Dimensão y_test:', y_test.shape)
```

```
[ ]  # Grid Search + Validação Cruzada para a Árvore de Decisão
     param_dict = {
         'criterion': ['gini','entropy'],
         'max_depth': range(1, 9),
         'min_samples_split': range(2, 9),
         'min_samples_leaf': range(1, 4)
     }

     grid = GridSearchCV(DecisionTreeClassifier(), param_grid=param_dict, cv=10, verbose=1, n_jobs=-1)
     grid.fit(X_train, y_train)

     print('\nMelhores parâmetros:', grid.best_params_)
     print('Melhor acurácia média na validação cruzada:', grid.best_score_)

     # Avaliação final com os melhores parâmetros
     best_model = DecisionTreeClassifier(**grid.best_params_)
     best_model.fit(X_train, y_train)
     y_pred = best_model.predict(X_test)

     # Matriz de Confusão
     cm = confusion_matrix(y_test, y_pred)
     labels_in_test = np.unique(y_test)
     cmd = ConfusionMatrixDisplay(cm, display_labels=labels_in_test)
     cmd.plot()
     plt.xticks(rotation=45, ha='right')
     plt.xlabel('Previsto')
     plt.ylabel('Verdadeiro')
     plt.title('Matriz de Confusão - Árvore de Decisão')
     plt.tight_layout()
     plt.show()
```

```
     # Initialize lists for metrics
     ac_trein = []
     ac_prev = []
     prec_cat1 = []
     prec_cat2 = []
     prec_cat3 = []
     prec_cat4 = []
     rec_cat1 = []
     rec_cat2 = []
     rec_cat3 = []
     rec_cat4 = []
     f1_cat1 = []
     f1_cat2 = []
     f1_cat3 = []
     f1_cat4 = []


     # Precisão por classe (handle potential division by zero with 1e-10)
     # Check if the column index exists in cm before accessing
     prec_cat1.append(cm[0,0] / (np.sum(cm[:,0]) + 1e-10) if cm.shape[1] > 0 else np.nan)
     prec_cat2.append(cm[1,1] / (np.sum(cm[:,1]) + 1e-10) if cm.shape[1] > 1 else np.nan)
     prec_cat3.append(cm[2,2] / (np.sum(cm[:,2]) + 1e-10) if cm.shape[1] > 2 else np.nan)
     prec_cat4.append(cm[3,3] / (np.sum(cm[:,3]) + 1e-10) if cm.shape[1] > 3 else np.nan)

     # Revocação por classe (handle potential division by zero with 1e-10)
     # Check if the row index exists in cm before accessing
     rec_cat1.append(cm[0,0] / (np.sum(cm[0,:]) + 1e-10) if cm.shape[0] > 0 else np.nan)
     rec_cat2.append(cm[1,1] / (np.sum(cm[1,:]) + 1e-10) if cm.shape[0] > 1 else np.nan)
     rec_cat3.append(cm[2,2] / (np.sum(cm[2,:]) + 1e-10) if cm.shape[0] > 2 else np.nan)
     rec_cat4.append(cm[3,3] / (np.sum(cm[3,:]) + 1e-10) if cm.shape[0] > 3 else np.nan)
```

```
# F1-score por classe (handle potential division by zero with 1e-10)
f1_cat1.append(2 * (prec_cat1[-1] * rec_cat1[-1]) / (prec_cat1[-1] + rec_cat1[-1] + 1e-10) if not np.isnan(prec_cat1[-1]) and not np.isnan(rec_cat1[-1]) else np.nan)
f1_cat2.append(2 * (prec_cat2[-1] * rec_cat2[-1]) / (prec_cat2[-1] + rec_cat2[-1] + 1e-10) if not np.isnan(prec_cat2[-1]) and not np.isnan(rec_cat2[-1]) else np.nan)
f1_cat3.append(2 * (prec_cat3[-1] * rec_cat3[-1]) / (prec_cat3[-1] + rec_cat3[-1] + 1e-10) if not np.isnan(prec_cat3[-1]) and not np.isnan(rec_cat3[-1]) else np.nan)
f1_cat4.append(2 * (prec_cat4[-1] * rec_cat4[-1]) / (prec_cat4[-1] + rec_cat4[-1] + 1e-10) if not np.isnan(prec_cat4[-1]) and not np.isnan(rec_cat4[-1]) else np.nan)
# Acurácia Final
acc = accuracy_score(y_test, y_pred)
print(f'\nAcurácia final no conjunto de teste: {acc:.2f}')

# Impressão dos resultados médios e desvios padrão (using nanmean and nanstd to ignore NaNs)
print('Acurácia de treinamento:', round(np.nanmean(ac_trein), 3), '+/-', round(np.nanstd(ac_trein), 3))
print('Acurácia de previsão:', round(np.nanmean(ac_prev), 3), '+/-', round(np.nanstd(ac_prev), 3))

print('\nPrecisão por categoria:')
print('C2:', round(np.nanmean(prec_cat1), 3), '+/-', round(np.nanstd(prec_cat1), 3))
print('C3:', round(np.nanmean(prec_cat2), 3), '+/-', round(np.nanstd(prec_cat2), 3))
print('C4:', round(np.nanmean(prec_cat3), 3), '+/-', round(np.nanstd(prec_cat3), 3))
print('O.G:', round(np.nanmean(prec_cat4), 3), '+/-', round(np.nanstd(prec_cat4), 3))

print('\nRecall por categoria:')
print('C2:', round(np.nanmean(rec_cat1), 3), '+/-', round(np.nanstd(rec_cat1), 3))
print('C3:', round(np.nanmean(rec_cat2), 3), '+/-', round(np.nanstd(rec_cat2), 3))
print('C4:', round(np.nanmean(rec_cat3), 3), '+/-', round(np.nanstd(rec_cat3), 3))
print('O.G:', round(np.nanmean(rec_cat4), 3), '+/-', round(np.nanstd(rec_cat4), 3))

print('\nF1-score por categoria:')
print('C2:', round(np.nanmean(f1_cat1), 3), '+/-', round(np.nanstd(f1_cat1), 3))
print('C3:', round(np.nanmean(f1_cat2), 3), '+/-', round(np.nanstd(f1_cat2), 3))
print('C4:', round(np.nanmean(f1_cat3), 3), '+/-', round(np.nanstd(f1_cat3), 3))
print('O.G:', round(np.nanmean(f1_cat4), 3), '+/-', round(np.nanstd(f1_cat4), 3))
```

```
# Importações necessárias
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.model_selection import train_test_split
import numpy as np
import matplotlib.pyplot as plt

# Listas para armazenar as métricas
ac_trein = []
ac_prev = []

prec_cat1 = []
prec_cat2 = []
prec_cat3 = []
prec_cat4 = []

rec_cat1 = []
rec_cat2 = []
rec_cat3 = []
rec_cat4 = []

f1_cat1 = []
f1_cat2 = []
f1_cat3 = []
f1_cat4 = []

# Get all unique labels from the target variable
all_labels = sorted(np.unique(y_dados))
```

```
# Loop de 100 execuções
for i in range(100):
    X_train, X_test, y_train, y_test = train_test_split(X_dados_p, y_dados, test_size=0.25)

    arv_dec = DecisionTreeClassifier(criterion='gini', max_depth=2, min_samples_leaf=1, min_samples_split=2) # Using best parameters from previous cell
    arv_dec.fit(X_train, y_train)
    y_pred = arv_dec.predict(X_test)

    # Generate confusion matrix with all possible labels
    cm = confusion_matrix(y_test, y_pred, labels=all_labels)

    # Acurácias
    ac_trein.append(arv_dec.score(X_train, y_train))
    ac_prev.append(accuracy_score(y_test, y_pred))

    # Precisão por classe (handle potential division by zero with 1e-10)
    # Check if the column index exists in cm before accessing
    prec_cat1.append(cm[0,0] / (np.sum(cm[:,0]) + 1e-10) if cm.shape[1] > 0 else np.nan)
    prec_cat2.append(cm[1,1] / (np.sum(cm[:,1]) + 1e-10) if cm.shape[1] > 1 else np.nan)
    prec_cat3.append(cm[2,2] / (np.sum(cm[:,2]) + 1e-10) if cm.shape[1] > 2 else np.nan)
    prec_cat4.append(cm[3,3] / (np.sum(cm[:,3]) + 1e-10) if cm.shape[1] > 3 else np.nan)

    # Revocação por classe (handle potential division by zero with 1e-10)
    # Check if the row index exists in cm before accessing
    rec_cat1.append(cm[0,0] / (np.sum(cm[0,:]) + 1e-10) if cm.shape[0] > 0 else np.nan)
    rec_cat2.append(cm[1,1] / (np.sum(cm[1,:]) + 1e-10) if cm.shape[0] > 1 else np.nan)
    rec_cat3.append(cm[2,2] / (np.sum(cm[2,:]) + 1e-10) if cm.shape[0] > 2 else np.nan)
    rec_cat4.append(cm[3,3] / (np.sum(cm[3,:]) + 1e-10) if cm.shape[0] > 3 else np.nan)
```

```
    # F1-score por classe (handle potential division by zero with 1e-10)
    f1_cat1.append(2 * (prec_cat1[-1] * rec_cat1[-1]) / (prec_cat1[-1] + rec_cat1[-1] + 1e-10) if not np.isnan(prec_cat1[-1]) and not np.isnan(rec_cat1[-1]) else np.nan)
    f1_cat2.append(2 * (prec_cat2[-1] * rec_cat2[-1]) / (prec_cat2[-1] + rec_cat2[-1] + 1e-10) if not np.isnan(prec_cat2[-1]) and not np.isnan(rec_cat2[-1]) else np.nan)
    f1_cat3.append(2 * (prec_cat3[-1] * rec_cat3[-1]) / (prec_cat3[-1] + rec_cat3[-1] + 1e-10) if not np.isnan(prec_cat3[-1]) and not np.isnan(rec_cat3[-1]) else np.nan)
    f1_cat4.append(2 * (prec_cat4[-1] * rec_cat4[-1]) / (prec_cat4[-1] + rec_cat4[-1] + 1e-10) if not np.isnan(prec_cat4[-1]) and not np.isnan(rec_cat4[-1]) else np.nan)


# Impressão dos resultados médios e desvios padrão (using nanmean and nanstd to ignore NaNs)
print('Acurácia de treinamento:', round(np.nanmean(ac_trein), 3), '+/-', round(np.nanstd(ac_trein), 3))
print('Acurácia de previsão:', round(np.nanmean(ac_prev), 3), '+/-', round(np.nanstd(ac_prev), 3))

print('\nPrecisão por categoria:')
print('C2:', round(np.nanmean(prec_cat1), 3), '+/-', round(np.nanstd(prec_cat1), 3))
print('C3:', round(np.nanmean(prec_cat2), 3), '+/-', round(np.nanstd(prec_cat2), 3))
print('C4:', round(np.nanmean(prec_cat3), 3), '+/-', round(np.nanstd(prec_cat3), 3))
print('O.G:', round(np.nanmean(prec_cat4), 3), '+/-', round(np.nanstd(prec_cat4), 3))

print('\nRecall por categoria:')
print('C2:', round(np.nanmean(rec_cat1), 3), '+/-', round(np.nanstd(rec_cat1), 3))
print('C3:', round(np.nanmean(rec_cat2), 3), '+/-', round(np.nanstd(rec_cat2), 3))
print('C4:', round(np.nanmean(rec_cat3), 3), '+/-', round(np.nanstd(rec_cat3), 3))
print('O.G:', round(np.nanmean(rec_cat4), 3), '+/-', round(np.nanstd(rec_cat4), 3))

print('\nF1-score por categoria:')
print('C2:', round(np.nanmean(f1_cat1), 3), '+/-', round(np.nanstd(f1_cat1), 3))
print('C3:', round(np.nanmean(f1_cat2), 3), '+/-', round(np.nanstd(f1_cat2), 3))
print('C4:', round(np.nanmean(f1_cat3), 3), '+/-', round(np.nanstd(f1_cat3), 3))
print('O.G:', round(np.nanmean(f1_cat4), 3), '+/-', round(np.nanstd(f1_cat4), 3))
```

```
# Salvando os arrays (se quiser analisar depois)
ac_prev_arv = np.copy(ac_prev)

prec_cat1_arv = np.copy(prec_cat1)
prec_cat2_arv = np.copy(prec_cat2)
prec_cat3_arv = np.copy(prec_cat3)
prec_cat4_arv = np.copy(prec_cat4)

rec_cat1_arv = np.copy(rec_cat1)
rec_cat2_arv = np.copy(rec_cat2)
rec_cat3_arv = np.copy(rec_cat3)
rec_cat4_arv = np.copy(rec_cat4)

f1_cat1_arv = np.copy(f1_cat1)
f1_cat2_arv = np.copy(f1_cat2)
f1_cat3_arv = np.copy(f1_cat3)
f1_cat4_arv = np.copy(f1_cat4)
```

```
import graphviz
from sklearn.tree import DecisionTreeClassifier, export_graphviz
import numpy as np

# Assuming 'best_model' is your trained DecisionTreeClassifier
# Assuming 'dados' is your original dataframe with feature names
# Assuming 'y_dados' is your target variable array

dot_data = export_graphviz(best_model,
                           out_file=None,
                           feature_names=dados.columns,
                           class_names=np.unique(y_dados), # Use unique values from y_dados as class names
                           filled=True,
                           rounded=True,
                           special_characters=True)

graph = graphviz.Source(dot_data)
graph.render("arvore_decisao", view=True)
```

```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestClassifier  # RandomForest está no sklearn.ensemble
from sklearn.metrics import confusion_matrix, accuracy_score, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

# Carregar os dados
# Exemplo: dados = pd.read_csv('caminho_para_dados.csv')
# Certifique-se de definir 'dados' e 'dados_cl' antes de executar este script

# Separar X (atributos preditores) e y (classe)
X_dados = dados.iloc[:, 0:15].values
y_dados = dados_cl['Classification 2'].values

print('Dimensão dos atributos preditores:', X_dados.shape)
print('Dimensão do atributo classe:', y_dados.shape)

# Padronização dos dados
pad = StandardScaler()
X_dados_p = pad.fit_transform(X_dados)

# Divisão em treino e teste
X_train, X_test, y_train, y_test = train_test_split(X_dados_p, y_dados, test_size=0.25, random_state=42)

print('\nDimensão X_train:', X_train.shape)
print('Dimensão X_test:', X_test.shape)
print('Dimensão y_train:', y_train.shape)
print('Dimensão y_test:', y_test.shape)
```

```python
# Parâmetros para o GridSearch
param = {
    'criterion': ['gini', 'entropy'],
    'n_estimators': [150, 250, 350],
    'min_samples_split': range(2, 5),  # range(1,5) causaria erro (mínimo é 2)
    'min_samples_leaf': range(1, 5)
}

# Grid Search com validação cruzada
G = GridSearchCV(estimator=RandomForestClassifier(random_state=3), param_grid=param, cv=10, verbose=1, n_jobs=-1)
G.fit(X_train, y_train)

print('Os melhores parâmetros são:', G.best_params_)
print('Acurácia média (validação cruzada):', G.best_score_)

# Utilizando os melhores parâmetros encontrados
melhores_parametros = G.best_params_

RF = RandomForestClassifier(
    n_estimators=melhores_parametros['n_estimators'],
    criterion=melhores_parametros['criterion'],
    min_samples_split=melhores_parametros['min_samples_split'],
    min_samples_leaf=melhores_parametros['min_samples_leaf'],
    random_state=3
)

RF.fit(X_train, y_train)
y_pred = RF.predict(X_test)
```

```python
# Matriz de Confusão
cm = confusion_matrix(y_test, y_pred)
labels_in_test = np.unique(y_test)
cmd = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=labels_in_test)
cmd.plot()
plt.xticks(rotation=45, ha='right')
plt.xlabel('Previsto')
plt.ylabel('Verdadeiro')
plt.title('Matriz de Confusão - Random Forest')
plt.tight_layout()
plt.show()

# Acurácia Final
acc = accuracy_score(y_test, y_pred)
print(f'\nAcurácia final no conjunto de teste: {acc:.2f}')
```

```python
# Importações necessárias
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.model_selection import train_test_split
import numpy as np
import matplotlib.pyplot as plt
```

```python
# Listas para armazenar as métricas
ac_trein = []
ac_prev = []

prec_cat1 = []
prec_cat2 = []
prec_cat3 = []
prec_cat4 = []

rec_cat1 = []
rec_cat2 = []
rec_cat3 = []
rec_cat4 = []

f1_cat1 = []
f1_cat2 = []
f1_cat3 = []
f1_cat4 = []

# Get all unique labels from the target variable
all_labels = sorted(np.unique(y_dados))

# Loop de 100 execuções
for i in range(100):
    X_train, X_test, y_train, y_test = train_test_split(X_dados_p, y_dados, test_size=0.25)
```

```python
    # Classificador Floresta Randômica com hiperparâmetros simples (pode ajustar com base no GridSearch)
    clf = RandomForestClassifier(
        n_estimators=150,
        criterion='gini',
        min_samples_split=2,
        min_samples_leaf=1,
        max_depth=5,
        random_state=i  # Para variar a semente a cada execução
    )

    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)

    # Matriz de confusão com todas as classes
    cm = confusion_matrix(y_test, y_pred, labels=all_labels)

    # Acurácias
    ac_trein.append(clf.score(X_train, y_train))
    ac_prev.append(accuracy_score(y_test, y_pred))

    # Precisão por classe
    prec_cat1.append(cm[0,0] / (np.sum(cm[:,0]) + 1e-10) if cm.shape[1] > 0 else np.nan)
    prec_cat2.append(cm[1,1] / (np.sum(cm[:,1]) + 1e-10) if cm.shape[1] > 1 else np.nan)
    prec_cat3.append(cm[2,2] / (np.sum(cm[:,2]) + 1e-10) if cm.shape[1] > 2 else np.nan)
    prec_cat4.append(cm[3,3] / (np.sum(cm[:,3]) + 1e-10) if cm.shape[1] > 3 else np.nan)

    # Revocação por classe
    rec_cat1.append(cm[0,0] / (np.sum(cm[0,:]) + 1e-10) if cm.shape[0] > 0 else np.nan)
    rec_cat2.append(cm[1,1] / (np.sum(cm[1,:]) + 1e-10) if cm.shape[0] > 1 else np.nan)
    rec_cat3.append(cm[2,2] / (np.sum(cm[2,:]) + 1e-10) if cm.shape[0] > 2 else np.nan)
    rec_cat4.append(cm[3,3] / (np.sum(cm[3,:]) + 1e-10) if cm.shape[0] > 3 else np.nan)
```

```python
    # F1-score por classe
    f1_cat1.append(2 * (prec_cat1[-1] * rec_cat1[-1]) / (prec_cat1[-1] + rec_cat1[-1] + 1e-10) if not np.isnan(prec_cat1[-1]) and not np.isnan(rec_cat1[-1]) else np.nan)
    f1_cat2.append(2 * (prec_cat2[-1] * rec_cat2[-1]) / (prec_cat2[-1] + rec_cat2[-1] + 1e-10) if not np.isnan(prec_cat2[-1]) and not np.isnan(rec_cat2[-1]) else np.nan)
    f1_cat3.append(2 * (prec_cat3[-1] * rec_cat3[-1]) / (prec_cat3[-1] + rec_cat3[-1] + 1e-10) if not np.isnan(prec_cat3[-1]) and not np.isnan(rec_cat3[-1]) else np.nan)
    f1_cat4.append(2 * (prec_cat4[-1] * rec_cat4[-1]) / (prec_cat4[-1] + rec_cat4[-1] + 1e-10) if not np.isnan(prec_cat4[-1]) and not np.isnan(rec_cat4[-1]) else np.nan)

# Impressão dos resultados médios e desvios padrão
print('Acurácia de treinamento:', round(np.nanmean(ac_trein), 3), '+/-', round(np.nanstd(ac_trein), 3))
print('Acurácia de previsão:', round(np.nanmean(ac_prev), 3), '+/-', round(np.nanstd(ac_prev), 3))

print('\nPrecisão por categoria:')
print('C2:', round(np.nanmean(prec_cat1), 3), '+/-', round(np.nanstd(prec_cat1), 3))
print('C3:', round(np.nanmean(prec_cat2), 3), '+/-', round(np.nanstd(prec_cat2), 3))
print('C4:', round(np.nanmean(prec_cat3), 3), '+/-', round(np.nanstd(prec_cat3), 3))
print('O.G:', round(np.nanmean(prec_cat4), 3), '+/-', round(np.nanstd(prec_cat4), 3))

print('\nRecall por categoria:')
print('C2:', round(np.nanmean(rec_cat1), 3), '+/-', round(np.nanstd(rec_cat1), 3))
print('C3:', round(np.nanmean(rec_cat2), 3), '+/-', round(np.nanstd(rec_cat2), 3))
print('C4:', round(np.nanmean(rec_cat3), 3), '+/-', round(np.nanstd(rec_cat3), 3))
print('O.G:', round(np.nanmean(rec_cat4), 3), '+/-', round(np.nanstd(rec_cat4), 3))

print('\nF1-score por categoria:')
print('C2:', round(np.nanmean(f1_cat1), 3), '+/-', round(np.nanstd(f1_cat1), 3))
print('C3:', round(np.nanmean(f1_cat2), 3), '+/-', round(np.nanstd(f1_cat2), 3))
print('C4:', round(np.nanmean(f1_cat3), 3), '+/-', round(np.nanstd(f1_cat3), 3))
print('O.G:', round(np.nanmean(f1_cat4), 3), '+/-', round(np.nanstd(f1_cat4), 3))
```

```python
    # Salvando os arrays (opcional)
    ac_prev_rf = np.copy(ac_prev)

    prec_cat1_rf = np.copy(prec_cat1)
    prec_cat2_rf = np.copy(prec_cat2)
    prec_cat3_rf = np.copy(prec_cat3)
    prec_cat4_rf = np.copy(prec_cat4)

    rec_cat1_rf = np.copy(rec_cat1)
    rec_cat2_rf = np.copy(rec_cat2)
    rec_cat3_rf = np.copy(rec_cat3)
    rec_cat4_rf = np.copy(rec_cat4)

    f1_cat1_rf = np.copy(f1_cat1)
    f1_cat2_rf = np.copy(f1_cat2)
    f1_cat3_rf = np.copy(f1_cat3)
    f1_cat4_rf = np.copy(f1_cat4)
```

```python
from scipy import stats
import numpy as np

# Teste para Acurácia
mann_ac = stats.mannwhitneyu(ac_prev_arv, ac_prev_rf, alternative = 'two-sided')
print('Mediana da acurácia de previsão com AD: ', np.median(ac_prev_arv), '\n')
print('Mediana da acurácia de previsão com FR: ', np.median(ac_prev_rf), '\n')
print('As medianas da acurácia de previsão, com 95% de confiança, foram consideradas: ')
if mann_ac[1] < 0.05:
  print('Significativamente diferentes, com p-valor igual a:', mann_ac[1])
else:
  print('Não havendo evidências contrárias, estima-se que são iguais, com p-valor igual a:', mann_ac[1])

print("-" * 30) # Separador

# Testes para Precisão por categoria
print("\nTestes para Precisão por categoria:")

# C2
mann_prec_c2 = stats.mannwhitneyu(prec_cat1_arv, prec_cat1_rf, alternative = 'two-sided')
print('Mediana da Precisão C2 com AD: ', np.median(prec_cat1_arv), '\n')
print('Mediana da Precisão C2 com FR: ', np.median(prec_cat1_rf), '\n')
print('As medianas da Precisão C2, com 95% de confiança, foram consideradas: ')
if mann_prec_c2[1] < 0.05:
  print('Significativamente diferentes, com p-valor igual a:', mann_prec_c2[1])
else:
  print('Não havendo evidências contrárias, estima-se que são iguais, com p-valor igual a:', mann_prec_c2[1])
print("-" * 10)
```

```python
# C3
mann_prec_c3 = stats.mannwhitneyu(prec_cat2_arv, prec_cat2_rf, alternative = 'two-sided')
print('Mediana da Precisão C3 com AD: ', np.median(prec_cat2_arv), '\n')
print('Mediana da Precisão C3 com FR: ', np.median(prec_cat2_rf), '\n')
print('As medianas da Precisão C3, com 95% de confiança, foram consideradas: ')
if mann_prec_c3[1] < 0.05:
  print('Significativamente diferentes, com p-valor igual a:', mann_prec_c3[1])
else:
  print('Não havendo evidências contrárias, estima-se que são iguais, com p-valor igual a:', mann_prec_c3[1])
print("-" * 10)

# C4
mann_prec_c4 = stats.mannwhitneyu(prec_cat3_arv, prec_cat3_rf, alternative = 'two-sided')
print('Mediana da Precisão C4 com AD: ', np.median(prec_cat3_arv), '\n')
print('Mediana da Precisão C4 com FR: ', np.median(prec_cat3_rf), '\n')
print('As medianas da Precisão C4, com 95% de confiança, foram consideradas: ')
if mann_prec_c4[1] < 0.05:
  print('Significativamente diferentes, com p-valor igual a:', mann_prec_c4[1])
else:
  print('Não havendo evidências contrárias, estima-se que são iguais, com p-valor igual a:', mann_prec_c4[1])
print("-" * 10)
```

```
# O.G.
mann_prec_og = stats.mannwhitneyu(prec_cat4_arv, prec_cat4_rf, alternative = 'two-sided')
print('Mediana da Precisão O.G. com AD: ', np.median(prec_cat4_arv), '\n')
print('Mediana da Precisão O.G. com FR: ', np.median(prec_cat4_rf), '\n')
print('As medianas da Precisão O.G., com 95% de confiança, foram consideradas: ')
if mann_prec_og[1] < 0.05:
  print('Significativamente diferentes, com p-valor igual a:', mann_prec_og[1])
else:
  print('Não havendo evidências contrárias, estima-se que são iguais, com p-valor igual a:', mann_prec_og[1])
print("-" * 30)


# Testes para Recall por categoria
print("\nTestes para Recall por categoria:")

# C2
mann_rec_c2 = stats.mannwhitneyu(rec_cat1_arv, rec_cat1_rf, alternative = 'two-sided')
print('Mediana do Recall C2 com AD: ', np.median(rec_cat1_arv), '\n')
print('Mediana do Recall C2 com FR: ', np.median(rec_cat1_rf), '\n')
print('As medianas do Recall C2, com 95% de confiança, foram consideradas: ')
if mann_rec_c2[1] < 0.05:
  print('Significativamente diferentes, com p-valor igual a:', mann_rec_c2[1])
else:
  print('Não havendo evidências contrárias, estima-se que são iguais, com p-valor igual a:', mann_rec_c2[1])
print("-" * 10)
```

```
# C3
mann_rec_c3 = stats.mannwhitneyu(rec_cat2_arv, rec_cat2_rf, alternative = 'two-sided')
print('Mediana do Recall C3 com AD: ', np.median(rec_cat2_arv), '\n')
print('Mediana do Recall C3 com FR: ', np.median(rec_cat2_rf), '\n')
print('As medianas do Recall C3, com 95% de confiança, foram consideradas: ')
if mann_rec_c3[1] < 0.05:
  print('Significativamente diferentes, com p-valor igual a:', mann_rec_c3[1])
else:
  print('Não havendo evidências contrárias, estima-se que são iguais, com p-valor igual a:', mann_rec_c3[1])
print("-" * 10)

# C4
mann_rec_c4 = stats.mannwhitneyu(rec_cat3_arv, rec_cat3_rf, alternative = 'two-sided')
print('Mediana do Recall C4 com AD: ', np.median(rec_cat3_arv), '\n')
print('Mediana do Recall C4 com FR: ', np.median(rec_cat3_rf), '\n')
print('As medianas do Recall C4, com 95% de confiança, foram consideradas: ')
if mann_rec_c4[1] < 0.05:
  print('Significativamente diferentes, com p-valor igual a:', mann_rec_c4[1])
else:
  print('Não havendo evidências contrárias, estima-se que são iguais, com p-valor igual a:', mann_rec_c4[1])
print("-" * 10)
```

```
# O.G.
mann_rec_og = stats.mannwhitneyu(rec_cat4_arv, rec_cat4_rf, alternative = 'two-sided')
print('Mediana do Recall O.G. com AD: ', np.median(rec_cat4_arv), '\n')
print('Mediana do Recall O.G. com FR: ', np.median(rec_cat4_rf), '\n')
print('As medianas do Recall O.G., com 95% de confiança, foram consideradas: ')
if mann_rec_og[1] < 0.05:
  print('Significativamente diferentes, com p-valor igual a:', mann_rec_og[1])
else:
  print('Não havendo evidências contrárias, estima-se que são iguais, com p-valor igual a:', mann_rec_og[1])
print("-" * 30)


# Testes para F1-score por categoria
print("\nTestes para F1-score por categoria:")

# C2
mann_f1_c2 = stats.mannwhitneyu(f1_cat1_arv, f1_cat1_rf, alternative = 'two-sided')
print('Mediana do F1-score C2 com AD: ', np.median(f1_cat1_arv), '\n')
print('Mediana do F1-score C2 com FR: ', np.median(f1_cat1_rf), '\n')
print('As medianas do F1-score C2, com 95% de confiança, foram consideradas: ')
if mann_f1_c2[1] < 0.05:
  print('Significativamente diferentes, com p-valor igual a:', mann_f1_c2[1])
else:
  print('Não havendo evidências contrárias, estima-se que são iguais, com p-valor igual a:', mann_f1_c2[1])
print("-" * 10)
```

```
# C3
mann_f1_c3 = stats.mannwhitneyu(f1_cat2_arv, f1_cat2_rf, alternative = 'two-sided')
print('Mediana do F1-score C3 com AD: ', np.median(f1_cat2_arv), '\n')
print('Mediana do F1-score C3 com FR: ', np.median(f1_cat2_rf), '\n')
print('As medianas do F1-score C3, com 95% de confiança, foram consideradas: ')
if mann_f1_c3[1] < 0.05:
  print('Significativamente diferentes, com p-valor igual a:', mann_f1_c3[1])
else:
  print('Não havendo evidências contrárias, estima-se que são iguais, com p-valor igual a:', mann_f1_c3[1])
print("-" * 10)

# C4
mann_f1_c4 = stats.mannwhitneyu(f1_cat3_arv, f1_cat3_rf, alternative = 'two-sided')
print('Mediana do F1-score C4 com AD: ', np.median(f1_cat3_arv), '\n')
print('Mediana do F1-score C4 com FR: ', np.median(f1_cat3_rf), '\n')
print('As medianas do F1-score C4, com 95% de confiança, foram consideradas: ')
if mann_f1_c4[1] < 0.05:
  print('Significativamente diferentes, com p-valor igual a:', mann_f1_c4[1])
else:
  print('Não havendo evidências contrárias, estima-se que são iguais, com p-valor igual a:', mann_f1_c4[1])
print("-" * 10)

# O.G.
mann_f1_og = stats.mannwhitneyu(f1_cat4_arv, f1_cat4_rf, alternative = 'two-sided')
print('Mediana do F1-score O.G. com AD: ', np.median(f1_cat4_arv), '\n')
print('Mediana do F1-score O.G. com FR: ', np.median(f1_cat4_rf), '\n')
print('As medianas do F1-score O.G., com 95% de confiança, foram consideradas: ')
if mann_f1_og[1] < 0.05:
  print('Significativamente diferentes, com p-valor igual a:', mann_f1_og[1])
else:
  print('Não havendo evidências contrárias, estima-se que são iguais, com p-valor igual a:', mann_f1_og[1])
print("-" * 30)
```