UNIVERSIDADE DO ESTADO DE SANTA CATARINA – UDESC
CENTRO DE CIÊNCIAS TECNOLÓGICAS - CCT
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA - PPGEEL

LUCAS LIMA NOGUEIRA

APLICAÇÃO DE ALGORITMOS DE APRENDIZAGEM DE MÁQUINA OTIMIZADOS NA PREVISÃO DE PROCESSOS TERMOQUÍMICOS E DE CAPACITÂNCIA DE SUPERCAPACITORES

JOINVILLE
2025

LUCAS LIMA NOGUEIRA

APLICAÇÃO DE ALGORITMOS DE APRENDIZAGEM DE MÁQUINA OTIMIZADOS NA PREVISÃO DE PROCESSOS TERMOQUÍMICOS E DE CAPACITÂNCIA DE SUPERCAPACITORES

Dissertação apresentada ao Programa de Pós–Graduação em Engenharia Elétrica do Centro de Ciências Tecnológicas da Universidade do Estado de Santa Catarina, como requisito parcial para a obtenção do grau de Mestre em Engenharia Elétrica.

Orientador: Ademir Nied

JOINVILLE
2025

Ficha catalográfica elaborada pelo programa de geração automática da
Biblioteca Universitária Udesc,
com os dados fornecidos pelo(a) autor(a)

Nogueira, Lucas Lima
APLICAÇÃO DE ALGORITMOS DE APRENDIZAGEM DE MÁQUINA OTIMIZADOS NA PREVISÃO DE PROCESSOS TERMOQUÍMICOS E DE CAPACITÂNCIA DE SUPERCAPACITORES / Lucas Lima Nogueira. -- 2025.
93 p.

Orientador: Ademir Nied
Dissertação (mestrado) -- Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação em Engenharia Elétrica, Joinville, 2025.

1. Supercapacitores. 2. Machine Learning. 3. Otimização. 4. Processos termoquímicos. I. Nied, Ademir. II. Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação em Engenharia Elétrica. III. Título.

LUCAS LIMA NOGUEIRA

APLICAÇÃO DE ALGORITMOS DE APRENDIZAGEM DE MÁQUINA OTIMIZADOS NA PREVISÃO DE PROCESSOS TERMOQUÍMICOS E DE CAPACITÂNCIA DE SUPERCAPACITORES

Dissertação apresentada ao Programa de Pós–Graduação em Engenharia Elétrica do Centro de Ciências Tecnológicas da Universidade do Estado de Santa Catarina, como requisito parcial para a obtenção do grau de Mestre em Engenharia Elétrica.

Orientador: Ademir Nied

Banca Examinadora:

Prof. Dr. Douglas Wildgrube Bertol
CCT/UDESC (Membro interno)

Dr. Rafael Ninno Muniz
UFF (Membro externo)

Prof. Dr. Stéfano Frizzo Stefenon
CCT/UDESC (Suplente interno)

Prof. Dr. Lucas Marques da Cunha
UNIR (Suplente externo)

Prof. Dr. Ademir Nied
CCT/UDESC (Presidente/Orientador)

Joinville, 15 de dezembro de 2025

AGRADECIMENTOS

Agradeço à minha família por todo o suporte durante o tempo de execução deste trabalho, sempre me apoiando e incentivando. Agradeço ao professor Ademir Nied, meu orientador, por ajudar e dar alternativas de como proceder ao longo do trabalho. Agradeço à secretaria do PPGEEL por sempre lembrar das obrigações ao longo do trabalho, agradeço especialmente à secretária Cíntia Tacla Gehring Fadel que sempre esteve disposta a ajudar, lembrando de prazos e demais detalhes ao longo da pesquisa. Agradeço aos professores André Bittencourt Leal e Douglas Wildgrube Bertol que atuaram como coordenadores do PPGEEL e sempre estiveram dispostos para dúvidas durante o período em que participei do programa.

Agradeço a Universidade do Estado de Santa Catarina (UDESC) e a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelas bolsas de mestrado durante este trabalho.

RESUMO

Por conta do crescimento na demanda de energia global, a necessidade de utilizar fontes renováveis de energia, além de usar eficientemente tais fontes, tem levado a uma crescente pesquisa em diferentes fontes de energia renováveis incluindo fontes que usam biomassa como matéria-prima, além de formas de armazenar essa energia mais eficientes. Com isso em mente, algoritmos de aprendizagem de máquina são utilizados para diminuir custos e tempos de teste/análise nesses campos de estudo. Este trabalho apresenta a utilização de algoritmos de aprendizagem de máquina (*Random Forest*, *CatBoost*, *eXtreme Gradient Boosting* e *Gradient Boosting Regressor*) em conjunto com otimização Bayesiana e otimização por enxame de partículas, na previsão de produtos finais em processos termoquímicos utilizando biomassa, obtendo resultados extremamente satisfatórios (com coeficiente de determinação entre 96% e 98%, obtendo 5% melhor precisão quando comparado a trabalho similar) e obtendo resultados satisfatórios na previsão de capacitância de um supercapacitor (simulando um novo material sendo estudado), mas obtendo baixa precisão (0,48 no índice R^2) no conjunto de teste (enquanto modelos similares testados neste trabalho obtiveram melhor índice R^2, mas resultaram em valores de regressão insatisfatórios).

Palavras-chave: Supercapacitores. *Machine Learning*. Otimização. Processos termoquímicos.

ABSTRACT

Due to the growth in global energy demand, the need to utilize renewable energy sources, as well as to use such sources efficiently, has led to increasing research into different renewable energy sources, including sources that use biomass as raw material, in addition to more efficient ways of storing this energy. With this in mind, machine learning algorithms are used to reduce costs and testing/analysis times in these fields of study. This work presents the use of machine learning algorithms (Random Forest, CatBoost, eXtreme Gradient Boosting, and Gradient Boosting Regressor) in conjunction with Bayesian optimization and particle swarm optimization, in the prediction of final products in thermochemical processes using biomass, obtaining extremely satisfactory results (with a coefficient of determination between 96% and 98%, achieving 5% better accuracy when compared to similar work) and obtaining satisfactory results in predicting the capacitance of a supercapacitor (simulating a new material being studied), but obtaining low accuracy (0.48 in the R^2 index) in the test set (while similar models tested in this work obtained a better R^2 index, but resulted in unsatisfactory regression values).

Keywords: Supercapacitors. Machine Learning. Optimization. Thermochemical processes.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

SC	Supercapacitor(es)
ML	*Machine Learning*
DL	*Deep Learning*
RF	*Random Forest (Floresta aleatória)*
XGBoost (XGB)	*eXtreme Gradient Boosting*
CART	*Classification and Regression Trees*
Bagging	*Bootstrap AGGregatING*
CV	*Cross-Validation*
KCV	*K-fold Cross-Validation*
BO	*Bayesian Optimization*

SUMÁRIO

1 INTRODUÇÃO

A demanda de energia mundial vem aumentando cada vez mais, sendo um dos motivos disso o aumento da população mundial estimada a chegar em 9,7 bilhões de pessoas em 2050 (Qubeissi; El-Kharouf, 2020). Consequentemente, a preocupação e busca por novas fontes de energia renovável e limpa aumenta proporcionalmente por conta da diminuição de energia fóssil disponível, danos ao meio ambiente por conta do uso de energia fóssil, aumento da demanda de energia global, possíveis problemas de transporte de energia (como ocorreu principalmente nos anos de 2022 e 2023 na Europa), dentre outros fatores (Mahmoudi-Qashqay et al., 2024), (Gajdzik et al., 2024), (Radhakrishnan et al., 2024), (Lobato-Peralta; Okoye; Alegre, 2024).

Uma alternativa ecológica de suprir essa demanda de energia nas atividades humanas é a utilização de energias renováveis. Esse tipo de energia possui menor quantidade de emissões de carbono, são mais limpas e o custo não é amarrado a flutuações que acontecem no mercado financeiro (Lobato-Peralta; Okoye; Alegre, 2024). As fontes de energia renovável mais estudadas incluem energia solar, energia eólica, energia advinda de biomassa e energia hidrelétrica. Entretanto, uma das principais desvantagens dessas fontes de energia são suas oscilações, a potência solar pode ser 14 vezes maior em um dia ensolarado quando comparado com um dia nublado, por exemplo (Sahani; Mahajan; Han, 2024), (Yakubu et al., 2024). Nesse contexto, há uma necessidade de melhorar ou substituir os métodos de armazenamento de energia atuais, prestando atenção à eficiência energética e à sustentabilidade, onde o desenvolvimento de supercapacitores (SC) vem sendo mais explorado nos últimos anos.

Infelizmente, há várias desvantagens de SCs, podendo-se citar: preços de mercado, potência durante curto período, baixa densidade de energia e alta absorção dielétrica. Um dos métodos mais explorados na tentativa de superar o obstáculo de baixa densidade de energia, por exemplo, é através do desenvolvimento de novos eletrodos para supercapacitores por conta que a funcionalidade e as características de supercapacitores originam-se do efeito mútuo dos eletrodos com os eletrólitos (Şahin; Blaabjerg; Sangwongwanich, 2022), (Iro; Subramani; Dash, 2016), (Nanda; Ghosh; Thomas, 2022). Para aumentar o desempenho de supercapacitores, é necessário entender a correlação entre características fundamentais do componente e sua respectiva capacitância, por exemplo, onde esse tipo de análise pode ser realizada por métodos baseados em modelo ou em dados, e estes são tópicos onde algoritmos de aprendizagem de máquina (do inglês, *Machine Learning* - ML) possuem excelentes resultados. A aplicação de algoritmos de aprendizagem de máquina permite o aprendizado através de dados experimentais passados e conhecimentos existentes para fazer previsões através de vários métodos em conjunto. Por conta da habilidade destes modelos de aumentar velocidade computacional, facilitar a compreensão de mecanismos complexos e otimizar o desempenho do objeto de estudo. Um modelo de ML criado com sucesso permite previsões que são baratas e precisas, possibilitando que pesquisadores identifiquem materiais com propriedades desejadas sem a necessidade de experimentos ou simulações que normalmente consomem muito mais dinheiro e/ou tempo

(Nanda; Ghosh; Thomas, 2022), (Sawant; Deshmukh; Awati, 2023), (Jha et al., 2023).

Com isso em mente, este trabalho apresenta a aplicação de métodos de aprendizagem de máquina em dois estudos diferentes: um dos estudos trata da previsão e otimização do bio-óleo em um processo termoquímico utilizando biomassa, enquanto o outro estudo trata da aplicação destes métodos na previsão da capacitância de supercapacitores. Métodos de limpeza de dados, otimização e os algoritmos de aprendizagem de máquina que fazem as previsões são combinados visando a obtenção de resultados satisfatórios.

O trabalho está estruturado da seguinte forma: o Capítulo 2 apresenta brevemente os conceitos utilizados durante a elaboração e utilização dos algoritmos de ML, o Capítulo 3 apresenta a metodologia utilizada durante a produção dos modelos de previsão utilizados, o Capítulo 4 apresenta os resultados e discussões acerca dos resultados obtidos, e por fim, o Capítulo 5 apresenta a conclusão do trabalho e as propostas de continuidade.

2 REVISÃO BIBLIOGRÁFICA

2.1 ENERGIA DE BIOMASSA

Energia atualmente é uma parte necessária da sociedade no desenvolvimento socioeconômico por conta da melhoria no padrão e qualidade de vida (Mirza; Ahmad; Majeed, 2008). Dentre as fontes de energia renovável e limpa, energia de biomassa está dentre as mais estudadas desde a década de 90. Em 1987, a utilização de biomassa como fonte de energia elétrica global já chegava a 14%, e nessa mesma época, a energia de biomassa começou a ser considerada como uma boa fonte de energia por conta de biomassa ter a capacidade de produzir eletricidade, combustíveis líquidos e gases (Hall, 1991). Em 2007, fontes de energia de biomassa já estavam entre as fontes de energia renovável mais promissoras, por conta de ter o potencial de aumentar a segurança energética em locais sem reservas abundantes de combustíveis fósseis, além da diminuição de emissões resultantes de carbono (Field; Campbell; Lobell, 2008) que é um problema extremamente importante devido ao aquecimento global atualmente.

Infelizmente, alguns estudos sugerem que energias renováveis tem um menor retorno de energia investida (Energy Return On Energy Invested - EROI), um menor EROI implica menos energia entregue para a sociedade relativa à energia necessária para fornecer a energia, ou seja, uma transição energética focando em menores quantidades de emissões de carbono pode resultar em menor quantidade de energia disponível para a sociedade (Slameršak; Kallis; O'Neill, 2022).

Além dessa preocupação com a energia disponível, nos últimos anos, outros fatores vêm contribuindo para fontes alternativas de energia. O conflito entre a Rússia e Ucrânia, por exemplo, afetou o mercado de energia por conta da alta dependência da Europa no fornecimento de recursos energéticos russos (Biniek-Poskart et al., 2023), inflação sem precedentes e preços de combustíveis fósseis cada vez mais altos com tendência para aumentar ao longo dos anos por conta que os recursos disponíveis estão rapidamente diminuindo e estudos sugerem que logo se esgotarão (Biniek-Poskart et al., 2023), (Khan et al., 2017).

Atualmente, é estimado que biomassa e resíduos representam aproximadamente 10% do fornecimento de energia global (de acordo com (Uddin et al., 2018), pessoas em áreas rurais em países em desenvolvimento, que representam 50% da população global, dependem de energia de biomassa) e é predito que a disponibilidade de biomassa tem capacidade de fornecer 1000% da energia necessária atual (Kan; Strezov; Evans, 2016). Entre as fontes de biomassa, a biomassa lignocelulósica[1] é a mais abundante, barata e mais amplamente encontrada no mundo, além de ser ecologicamente viável, sendo madeira a forma de biomassa lignocelulósica mais famosa. Comparando com outras biomassas (por exemplo: resíduos de animais, resíduos de processamento de comida, lodo de esgoto, resíduos sólidos municipais e resíduos de processos de agricultura), a biomassa lignocelulósica tem as vantagens de ter um menor teor de água, menor teor de cinzas, e composição mais simples (de forma que facilita a decomposição), além

[1] Material lignocelulósico consiste de principalmente três tipos diferentes de polímeros encontrados na natureza, especificamente celulose, hemicelulose e lignina, que são associados entre si (Hendriks; Zeeman, 2009).

de que o biocarvão produzido por biomassa lignocelulósica é melhor que o biocarvão de outras fontes em termos de custo, vantagens de estruturas superficiais (área superficial, por exemplo) e potenciais aplicações (Qin et al., 2022), (Qubeissi; El-Kharouf, 2020), (Fahmy et al., 2020).

2.2 PROCESSOS DE CONVERSÃO TERMOQUÍMICA

Existem vários tipos de processos de conversão termoquímica, podendo-se citar: combustão, pirólise, carbonização, torrefação, co-combustão, gaseificação e liquefação. A pirólise é considerada como o início de todos os processos de conversão termoquímica por conta que envolve todas as reações químicas (além de ter grande potencial por usar material não processado para produzir energia (Uddin et al., 2018)), geralmente produzindo submateriais nas três fases da matéria: resíduo sólido rico em carbono (chamado de carvão ou biocarvão) que pode conter impurezas (compostos aromáticos, por exemplo); produtos líquidos (industrialmente e economicamente conhecidos como bio-óleo e alcatrão), incluindo compostos aromáticos, fenóis, aldeídos e água, por exemplo; e produtos gasosos, incluindo monóxidos de carbono e hidrogênio (industrialmente conhecido como gás de síntese), metano, dióxido de carbono e hidrocarbonetos leves, por exemplo (Qubeissi; El-Kharouf, 2020), (Uddin et al., 2018), (Fahmy et al., 2020). Os gases e o bio-óleo são formados com o uso da fração volátil da biomassa, enquanto o biocarvão é formado com componente de carbono fixo (Qubeissi; El-Kharouf, 2020). É interessante mencionar que o alcatrão e o bio-óleo são bem relacionados, e algumas diferenças entre os dois líquidos incluem: o bio-óleo é formado por compostos de menor peso molecular que o alcatrão, o bio-óleo possui menor viscosidade, o alcatrão é um líquido preto ou marrom escuro. Rigorosamente, não existe uma definição amplamente utilizada do alcatrão, alguns estudos consideram alcatrão como todos os produtos pirolíticos de peso molecular maior que o benzeno (Fahmy et al., 2020).

A pirólise inicia-se com o aquecimento de material orgânico sendo utilizado através de um processo sem oxigênio ou em condições que limitam o oxigênio, sob temperaturas menores que 700° C (Kan; Strezov; Evans, 2016),(Qubeissi; El-Kharouf, 2020), (Akhtar; Krepl; Ivanova, 2018) (em condições limitantes de oxigênio, combustão ocorre em uma pequena parte da biomassa de forma que o calor liberado na combustão é usada para manter a temperatura do reator constante), se o material tiver úmido, durante o aquecimento do material, a temperatura aumenta localmente, levando primeiro a uma evaporação da umidade (estágio de secagem) e em seguida para uma liberação progressiva dos voláteis pirolíticos (primeiro estágio da pirólise). Os voláteis primários são produzidos pela cisão térmica de ligações químicas nos constituintes individuais da biomassa (que são celulose, hemicelulose, lignina e extrativos), onde esses voláteis primários incluem espécies de gases permanentes (CO_2, CO e CH_4, por exemplo) e de gases condensáveis (vários compostos orgânicos e água, por exemplo) em condições ambiente. Mesmo que os constituintes da biomassa se decomponham mais rapidamente em diferentes faixas de temperatura, normalmente, o primeiro estágio da pirólise finaliza em temperaturas menores que 500 °C. Se o material sendo utilizado é convertido a temperaturas mais altas, alguns dos

voláteis primários liberados dentro do material podem também participar em várias reações secundárias para formar mais subprodutos através de processos como *cracking*, recombinação, etc. O *cracking*, por exemplo, forma moléculas com menor peso molecular através da quebra de compostos voláteis (Kan; Strezov; Evans, 2016), (Uddin et al., 2018).

A complexidade da pirólise de biomassa aparece devido à diferença da decomposição dos componentes da biomassa com mecanismos e taxas de reação variantes que também dependem parcialmente das condições de processamento térmico e do *design* do reator (Kan; Strezov; Evans, 2016). A distinção entre pirólises primária e secundária não é perfeita, por conta que as reações voláteis secundárias podem ocorrer tanto nos poros da partícula e/ou no gás a granel, de forma que as reações primária e secundária podem ocorrer em diferentes partes da partícula ao mesmo tempo. O processo de formação do carvão inicia-se com a formação de anéis de benzeno, esses anéis então se combinam em um resíduo sólido conhecido como carvão que é uma estrutura policíclica aromática (durante este processo, água ou gás incondensável podem ser liberados), esse carvão resultante do estágio primário da pirólise pode também ser ativo durante as reações secundárias, catalizando a conversão de vapores orgânicos em gases leves (reações de quebra) e em carvão secundário (reações de polimerização), além do mais, o próprio carvão pode ser convertido em espécies gasosas através de reações de gaseificação com H_2O e CO_2, por exemplo (Neves et al., 2011), (Uddin et al., 2018).

A pirólise pode ser classificada de acordo com a taxa de aquecimento aplicada e o tempo de residência, podendo basicamente ser dividida em três tipos/métodos principais incluindo pirólise lenta ou convencional (menores taxas de aquecimento, resultando em maiores rendimentos de sólidos), pirólise rápida (taxa de aquecimento muito alta e tempo de residência menor) e pirólise *flash* (ou ultra rápida). A pirólise lenta pode ainda ser classificada em carbonização bem como torrefação (dependendo da temperatura de operação do processo e tempo de residência). O processo de carbonização utiliza maior temperatura de operação e maiores tempos de residência que a torrefação, enquanto o objetivo da carbonização é produzir um produto altamente carbonáceo, a torrefação poderia ser um pré-tratamento para processamentos adicionais. As proporções dos 3 subprodutos principais (biocarvão, bio-óleo e gás de síntese) da pirólise dependem do método da pirólise e dos parâmetros da reação (condições do reator, por exemplo) (Qubeissi; El-Kharouf, 2020), (Kan; Strezov; Evans, 2016), (Uddin et al., 2018). O bio-óleo produzido através do processo de pirólise muitas vezes pode ser utilizado efetivamente no lugar de combustíveis à base de hidrocarbonetos como o carvão, gás natural e petróleo, por exemplo. O biocarvão também pode possuir grande potencial como combustível alternativo ao carvão mineral, além da possibilidade de ser utilizado em gerenciamento de resíduos e remedição de solo, enquanto o gás pirolítico pode ser uma boa fonte de gás de síntese (podendo ser utilizado na produção de diesel, ácido acético, etc.) (Yu; Odriozola; Reina, 2019).

Gaseificação é uma das técnicas de conversão térmica mais amplamente utilizadas, realizada em temperaturas acima de 700° C (Akhtar; Krepl; Ivanova, 2018), responsável por converter os resíduos de biomassa em gás de síntese (uma mistura de H_2, CO_2, CH_4 e CO) e

carvão. Gás de síntese proveniente do processo de gaseificação pode ser usado para aquecimento e geração de energia. H_2 como o principal componente no gás de síntese, não é apenas o portador mais importante na geração de energia, mas também uma importante matéria-prima na indústria química para hidrogeração e hidrotratamento para sintetizar produtos valiosos, como o metanol, amônia e etilenoglicol. Além de produzir energia e produtos químicos valiosos, tais processos em alta temperatura conseguem também eficientemente imobilizar e eliminar produtos inorgânicos e poluentes orgânicos (Li et al., 2023). Infelizmente, alcatrão é um dos subprodutos que podem ser gerados no processo de gaseificação e que pode levar o processo de gaseificação à falha, sendo que atualmente ainda é um desafio encontrar uma abordagem eficiente e econômica para evitar a geração de alcatrão, devido à dificuldade de entender totalmente o processo de gaseificação. A distribuição dos subprodutos finais e a composição do gás de síntese são extremamente dependentes das propriedades da biomassa e das condições de gaseificação (de forma similar ao processo de pirólise). Por conta disso, métodos de aprendizagem de máquina vêm sendo utilizados para modelagem e interpretação dos processos de conversão de biomassa e fornecer percepções para otimizar as condições operacionais tanto na gaseificação quanto na pirólise ou copirólise (Dong et al., 2023), (Li et al., 2023).

2.2.1 Pirólise e copirólise de lodo de esgoto

Para exemplificar sobre a pirólise e introduzir o que é a copirólise de um dos materiais mais utilizados nesses tipos de processos, essa subseção introduzirá o uso do lodo de esgoto nesses processos. O processo de pirólise do lodo de esgoto possui desvantagens como a alta quantidade de cinzas e teor de umidade, resultando em produtos de baixa qualidade com altos valores de gás carbônico no gás, condensação do bio-óleo e biocarvão rico em cinzas. Além do mais, o óleo pirolítico advindo do lodo de esgoto contém hidrocarbonetos, água, oxigênio e compostos de nitrogênio, de forma que quando este óleo é diretamente utilizado como combustível pode resultar em instabilidade e emissões de NO_X e SO_X (óxidos de nitrogênio e enxofre, respectivamente), substâncias particularmente nocivas que além de serem precursores da chuva ácida (ameaçando a estabilidade ambiental e a sustentabilidade das reservas de alimentos e madeira), estas substâncias causam danos aos tecidos pulmonares e impactos negativos à saúde humana (Ma et al., 2023), (Singh; Agrawal, 2007). Para citar um dos estudos que obtiveram resultados interessantes utilizando o processo de copirólise, comenta-se no próximo parágrafo sobre uma pesquisa realizada na China.

A copirólise do lodo de esgoto emerge então como uma potente estratégia para enfrentar desafios (efeitos negativos apresentados no fim do parágrafo anterior, por exemplo), mostrando efeitos sinérgicos que melhoram a qualidade dos subprodutos, possui menor custo e é mais sustentável do que a pirólise do lodo de esgoto, tendo um maior rendimento de teor de hidrocarboneto, taxas de quebra mais rápidas e uma menor faixa de temperaturas (Ma et al., 2023). Produtos recicláveis que possuem altos valores que podem ser obtidos do processo de copirólise de lodo de esgoto com resíduo de caule de tabaco, por exemplo, incluem o ácido *ethylxanthogenacetic*

(utilizado na polimerização por transferência reversível de cadeia por adição-fragmentação). Esse tipo de polimerização pode ser usado no *design* de polímeros de arquiteturas complexas (Nguyen et al., 2008)), como o esqualeno (do inglês, *squalene/supraene*), os efeitos que se sobressaem do escaleno são as possibilidade de inibir câncer, inibir tumores e ação antioxidante na pele (Lozano-Grande et al., 2018)), o propeno ou propileno (utilizado como combustível em vários processos industriais, bem como na fabricação de cadeiras plásticas, brinquedos, etc.), e o 4-pentenal e ácido acético (utilizado na fabricação de vinagre, plásticos, corantes, inseticidas, produtos químicos para fotografias, etc) (Ma et al., 2023), (CETESB, 2012).

2.3 SUPERCAPACITOR

Os problemas e preocupações citados na seção anterior também instigaram o aumento de estudos sobre dispositivos alternativos de armazenamento de energia. Infelizmente, baterias e capacitores que são os dispositivos mais utilizados para armazenamento de energia elétrica são inadequados para altas densidades de potência e energia, características necessárias para o consumo efetivo e desempenho de sistemas de energias renováveis (Maksoud et al., 2021), além de que alguns dos dispositivos de armazenamento convencionais contém contaminantes que são prejudiciais para o meio ambiente (Şahin; Blaabjerg; Sangwongwanich, 2022). Nesse contexto, cientistas têm estudado supercapacitores como uma solução alternativa para aplicações híbridas ou individuais de armazenamento de energia elétrica (Şahin; Blaabjerg; Sangwongwanich, 2022). Supercapacitores (SCs) eletroquímicos ou capacitores elétricos de dupla camada (EDLC - *Electrical Double Layer Capacitor*) ou ultracapacitores têm sido altamente considerados por conta da sua rápida capacidade de armazenamento (baixo tempo de descarga de um SC que é de 1-10s vs. baterias de íons de lítio: 10-60min), estabilidade cíclica aprimorada (SC > 30.000h vs. bateria > 500h), baixa resistência interna (aproximadamente 0.01 Ω), alta densidade de potência, empacotamento flexível, larga faixa térmica (-40°C a 70°C), baixa necessidade de manutenção, baixo peso, além de ser ambientalmente amigável (Raza et al., 2018),(Iro; Subramani; Dash, 2016), (Şahin; Blaabjerg; Sangwongwanich, 2022),(Şahin; Blaabjerg, 2020). Algumas desvantagens de SC são: tempo de auto-descarga muito alta (aproximadamente entre 1 e 2 dias), conexões em série são necessárias para se obter uma maior tensão, fornece potência por um período muito curto e alta absorção dielétrica (Şahin; Blaabjerg, 2020).

Mesmo que muitos avanços tenham sido feitos no estudo de SCs, podendo-se citar diferentes tipos de materiais avançados e novos *designs*, SCs ainda possuem baixa densidade de energia de aproximadamente 8-30 Wh/L quando comparado com 500 Wh/L para baterias de lítio, por isso que em aplicações onde alta potência e vida útil aprimorada são requeridas como em veículos híbridos, guindastes, trens e elevadores, a combinação de baterias com SCs torna-se uma ótima solução (Zhao; Burke, 2021),(Raza et al., 2018).

Diferentemente da estrutura de um capacitor cerâmico ou eletrolítico, um SC consiste de dois eletrodos sólidos polarizados por uma tensão aplicada, e separados por uma membrana

separadora (que previne o contato entre os eletrodos positivo e negativo) e um eletrólito líquido (Şahin; Blaabjerg; Sangwongwanich, 2022), (Şahin; Blaabjerg, 2020). A superioridade de capacitância dos EDLCs sobre capacitores eletrolíticos e cerâmicos consiste do dielétrico mais fino e da alta área superficial dos eletrodos, sendo essas duas características também importantes para maior densidade de potência quando comparando com baterias, e maior densidade de energia quando comparando com capacitores (Prasad et al., 2019). Esses eletrodos, geralmente feitos de carbono ativado, tecido de fibra de carbono, grafite, grafeno e nanotubos de carbono, são posicionados nos coletores de eletricidade e, normalmente, revestidos com pó de carbono ativado. Dessa forma, uma dupla camada elétrica é gerada em cada interface onde o pó de carbono ativado toca o eletrólito. Quando o SC está carregado, os íons negativos e lacunas no lado do eletrodo positivo, e os íons positivos e os elétrons no lado do eletrodo negativo são alinhados através da interface, e esse alinhamento no SC forma o que é chamado EDLC, como pode ser visto na Figura 1 (Şahin; Blaabjerg, 2020), (Şahin; Blaabjerg; Sangwongwanich, 2022). Uma característica são os gráficos de carga e descarga de um SC que exibe uma curva galvanostática (GCD - *Galvanostatic Charge Discharge*) de formato quase triangular e uma curva de voltametria cíclica (CV - *Cyclic Voltammogram*) quase retangular devido à natureza fundamental do mecanismo de armazenamento do capacitor elétrico de dupla camada (Zhao; Burke, 2021). Devido sua estrutura, biocarvão de diferentes fontes pode ser utilizado como materiais eletrodos para supercapacitores, ou seja, um processo que começa utilizando resíduos orgânicos pode gerar supercapacitores no final (Khedulkar et al., 2023). Infelizmente, este trabalho não se estende por esse caminho por conta do tempo que seria necessário para tais análises, dito isso, a pesquisa acaba tendo foco no gás de síntese por simples escolha.

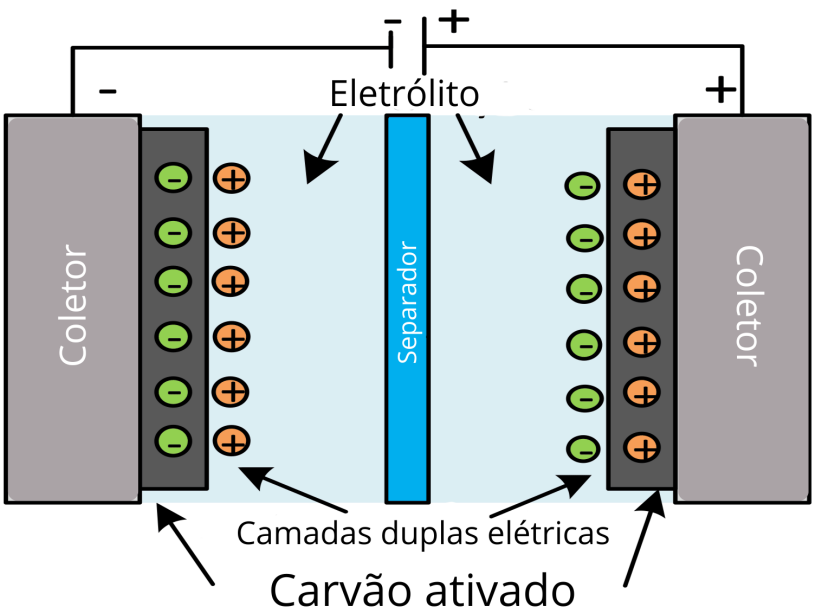

Figura 1 – Estrutura de um supercapacitor. Adaptado de (Şahin; Blaabjerg, 2020).

2.4 POR QUE UTILIZAR ALGORITMOS DE INTELIGÊNCIA ARTIFICIAL?

Atualmente, algoritmos de inteligência artificial são utilizados em problemas de diversas áreas diferentes, mais especificamente, algoritmos de aprendizagem de máquina (*Machine Learning* - ML) e aprendizagem profunda (*Deep Learning* - DL). Esses algoritmos têm demonstrado sucesso no *design* de materiais para armazenamento de energia (várias propriedades eletroquímicas como capacitância, densidade de potência ou durabilidade cíclica têm sido preditas com ótima precisão (Jitapunkul et al., 2023)), com suas predições sendo experimentalmente afirmadas pelos pesquisadores através de metodologias *design-to-device* (Mendhe; Panda, 2023). Utilizando algoritmos de ML, pesquisadores podem rapidamente analisar uma enorme quantidade de eletrodos sem a necessidade de experimentação de quantidades absurdas de materiais, além da enorme quantidade de tentativa e erro, acelerando o processo de descoberta de materiais, auxiliando na compreensão das propriedades do material, além de salvar tempo e recursos associados com experimentação (Ravichandran et al., 2024), (Sawant et al., 2025). Em (Deebansok et al., 2024), é comentada a dificuldade da interpretação direta de dados gráficos retirados de figuras usando métodos de mineração de dados atuais (2024) por conta que a aplicação de algoritmos de *machine learning* na análise de sinais eletroquímicos no campo de armazenamento de energia ainda está em fase inicial. O autor apresenta então uma metodologia baseada em classificação de imagem utilizando *machine learning* para analisar sinais eletroquímicos (mais especificamente, CV e GCD), sendo essa metodologia bem adequada também para dados sem *label*, sendo tolerante a ruídos e é capaz de utilizar dados complexos, conseguindo determinar o comportamento de eletrodos de milhares de artigos científicos. Em (Saramolee et al., 2025), uma rede emulada, adaptativa, para múltiplas entradas, com regras *fuzzy* foi utilizada para auxiliar a compreensão de quais propriedades de esponjas de biocarbono deveriam ser mais focadas, concluindo que a arquitetura dos poros e a condutividade elétrica tenderam a ter um maior efeito na performance capacitiva, além de afirmar que as esponjas de biocarbono poderiam ser boas candidatas a materiais para supercapacitores. Em (Mendhe; Panda, 2023), afirma-se que estudos também utilizam grande quantidade de dados com modelos de ML para verificar correlações entre importantes atributos de eletrodos e seus respectivos efeitos. Em (Ravichandran et al., 2024), os algoritmos *Artificial Neural Networks* (ANN), *Random Forest* (RF) e *XGBoost* (XGB) foram utilizados em um conjunto de dados de treino de mais de 200.000 linhas para fazer previsão de capacitância específica e corrente em diferentes concentrações de dopagem com diferentes materiais de dopagem, obtendo uma precisão média de 97,65%. Em (Yang et al., 2023), é feito um estudo sobre impressão 3D (tridimensional) de microrredes (*microlattice*) de carbono para supercapacitores utilizando RF, obtendo precisão de 97,8%, com o modelo de ML sugerindo que um aumento na espessura do eletrodo, diminuição da distância do *gap* e diminuição da área geométrica são favoráveis à performance de capacitância superficial. Análises adicionais indicaram que esses 3 parâmetros também afetam a proporção da superfície acessível do eletrodo para capacidade de armazenamento de energia e distribuição de corrente.

2.5 ALGORITMOS DE APRENDIZAGEM DE MÁQUINA (*MACHINE LEARNING* - ML)

Métodos de ML estão entre os mais importantes responsáveis de avanços recentes em matemática aplicada, com resultados significantes em problemas de classificação (onde a variável alvo/dependente é categórica, onde variável categórica é uma variável em que as unidades de observações são diferentes em termos de tipo ou espécie, como em uma variável que representa o gênero, por exemplo (Alkharusi, 2012)) e em problemas de regressão (onde a variável alvo é numérica), como comentado anteriormente, métodos de ML estão sendo cada vez mais utilizados em uma variedade de indústrias, incluindo assistência médica, finanças, engenharia, etc (Barboza; Kimura; Altman, 2017), (Sharifani; Amini, 2023), (Marcelino et al., 2021). Os algoritmos aqui utilizados foram selecionados após verificar-se resultados positivos em diversas literaturas.

2.5.1 Árvore de decisão - Decision Tree (*DT*)

O primeiro algoritmo a ser explicado será o de árvores de decisão, esse é o algoritmo utilizado como base na construção dos outros algoritmos utilizados nesta dissertação, tais como RF, XGBoost, CatBoost e GBR. Antes de explicar como uma árvore de decisão funciona,são apresentadas as definições dadas em Louppe (2014):

- **Definição 1**: Uma árvore é um grafo $G = (V, E)$ no qual quaisquer dois vértices (ou nós) são conectados por exatamente um caminho.
- **Definição 2**: Uma árvore enraizada é uma árvore na qual um dos nós foi designado como a raiz.
- **Definição 3**: Se existe uma aresta de t_1 para t_2 (isto é, se $(t_1,t_2) \in \mathcal{E}$), então o nó t_1 é dito ser "pai"do nó t_2, enquanto o nó t_2 é dito ser “filho” do nó t_1.
- **Definição 4**: Em uma árvore enraizada, um nó é chamado de interno se ele tem um ou mais “filhos” e é chamado de terminal se ele “não tem filhos”. Nós terminais são também conhecidos como folhas.

2.5.1.1 Árvores de classificação (classification trees)

No problema de classificação geral, é conhecido que cada instância em um conjunto de dados pertence a um número finito de classes possíveis, e dado um conjunto de medidas (valores das variáveis independentes/variáveis de entrada) para um caso, é desejado prever corretamente para qual classe o caso pertence. Inicia-se essa seção com árvores de classificação simplesmente por conta de tantas ideias-chave originarem desse algoritmo (Loh, 2014), (Sutton, 2005). Formalmente, um modelo de classificação é basicamente uma regra que atribui uma classe prevista baseado em um conjunto de valores relacionados, $x_1, x_2, ..., x_{K-1}$ e x_K. Considerando o espaço de variáveis independentes $\mathcal{X}$ para ser o conjunto de todos os possíveis valores de

$(x_1,...,x_K)$, e admitindo $\mathcal{C} = \{c_1, c_2, ..., c_J\}$ ser o conjunto de classes possíveis, um modelo de classificação é apenas uma função com domínio $\mathcal{X}$ e imagem $\mathcal{C}$, e corresponde para uma partição de $\mathcal{X}$ em conjuntos disjuntos, $B_1, B_2, ..., B_J$, tal que a classe prevista é j se $\mathbf{x} \in B_j$, onde $\mathbf{x} = (x_1,...,x_K)$, ou seja, cada B_j representa conjuntos de instâncias para cada classe j possível. Como o modelo de classificação usa valores anteriores como uma base para novas previsões, esse tipo de modelo é similar ao modelo de regressão, apenas o tipo de resposta é diferente, sendo nominal para a classificação e numérica para a regressão (Sutton, 2005).

Modelos de classificação estruturados em árvore são construídos fazendo divisões repetitivas tanto de $\mathcal{X}$ quanto dos subconjuntos de $\mathcal{X}$ criados, de forma que uma estrutura hierárquica é criada. Por exemplo, $\mathcal{X}$ poderia primeiro ser dividido em $\{\mathbf{x}|x_3 \leq 53,5\}$ e $\{\mathbf{x}|x_3 > 53,5\}$. Então, o primeiro destes conjuntos poderia ser dividido formando os seguintes conjuntos de valores $A_1 = \{\mathbf{x}|x_3 \leq 53,5, x_1 \leq 29,5\}$ e $A_2 = \{\mathbf{x}|x_3 \leq 53,5, x_1 > 29,5\}$, e o outro conjunto pode ser dividido em $A_3 = \{\mathbf{x}|x_3 > 53,5, x_1 \leq 74,5\}$ e $A_4 = \{\mathbf{x}|x_3 > 53,5, x_1 > 74,5\}$. A Figura 2 mostra a árvore com tais divisões.

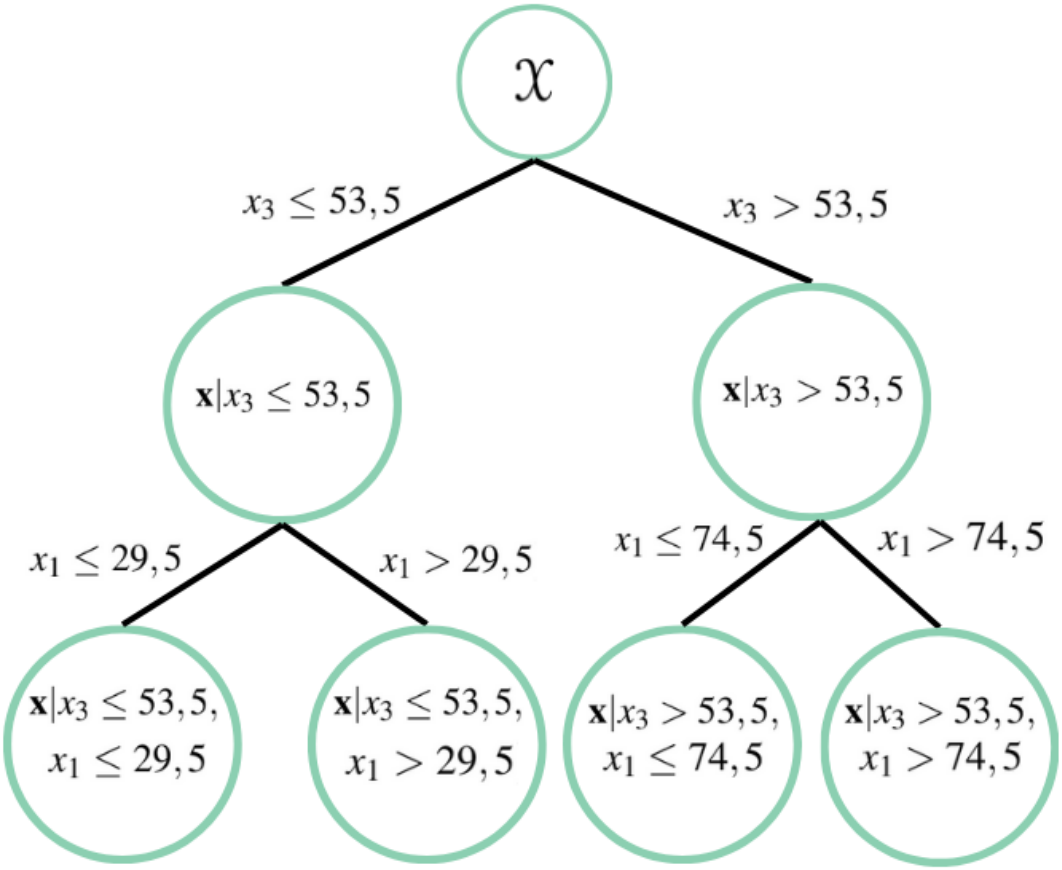

Figura 2 – Exemplo de uma árvore de decisão (autoria própria).

Se houverem apenas duas classes (J = 2), poderia ser que os casos possuindo classe desconhecida onde $\mathbf{x}$ pertence aos conjuntos A_1 ou A_3 deveriam ser classificados como c_1

(previstos para serem da classe c_1), e casos onde $\mathbf{x}$ pertence ao conjunto A_2 ou A_4 deveriam ser classificados como c_2. Fazendo uso da notação criada acima, teria-se $B_1 = A_1 \cup A_3$ e $B_2 = A_2 \cup A_4$, ou seja, a região B_1 seria composta das instâncias definidas pelos conjuntos A_1 e A_3 representando que aquelas instâncias foram classificadas como classe c_1, por exemplo (Sutton, 2005). Enquanto pode ser difícil mostrar o particionamento de $\mathcal{X}$ quando mais de duas variáveis de entrada são utilizadas, pode ser facilmente criada uma representação em árvore do modelo de classificação, que é fácil para usar não importa quantas variáveis são usadas e quantos conjuntos compõem o particionamento. Geralmente, é difícil entender como as variáveis preditoras (independentes) se relacionam às várias classes quando o modelo de classificação em árvore é muito complexo então o modelo de classificação pode ser utilizado facilmente, sem a necessidade de um computador, para classificar uma nova observação baseada nos valores de entrada, o que normalmente não é o caso para outros algoritmos, como classificadores *nearest neighbors* e baseados em *kernel*, por exemplo (Sutton, 2005). A Figura 3 mostra um exemplo da divisão de $\mathcal{X}$ (em um conjunto de dados de supercapacitores) em subconjuntos, onde TP = Tamanho dos Poros, AS = Área Superficial BET e ME = Material dos Eletrodos.

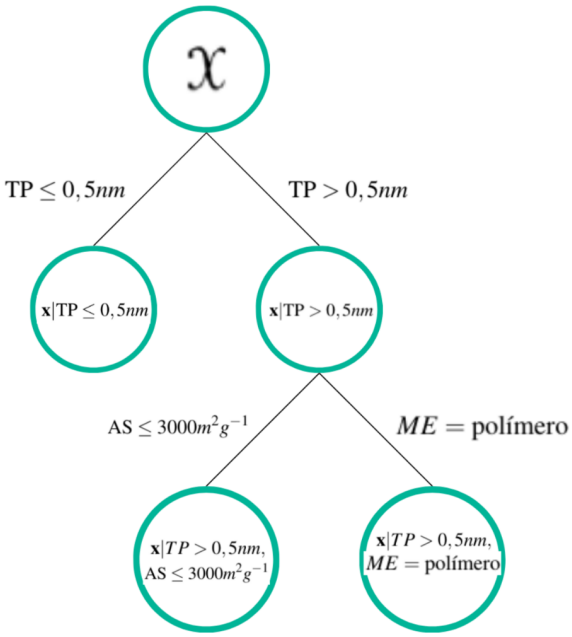

Figura 3 – Exemplo de uma árvore de decisão em um conjunto de dados de supercapacitores (autoria própria).

Deveria ser notado que quando $\mathcal{X}$ é dividido em 2 subconjuntos, estes subconjuntos não precisam ser divididos usando a mesma variável (como feito no primeiro exemplo, em que as regiões de x_3 em $\leq 53,5$ e $> 53,5$ foram utilizadas para dividir $\mathcal{X}$). Por exemplo, um subconjunto poderia ser dividido com base em x_1 e o outro subconjunto pode ser dividido com base em x_4. De forma que os princípios de classificação para diferentes regiões de $\mathcal{X}$ podem usar diferentes variáveis (como pode ser observado na Figura 3) ou serem diferentes funções das mesmas variáveis. Além do mais, uma árvore de classificação não tem que ser simétrica no

padrão de seus nós, ou seja, quando $\mathcal{X}$ é dividido, pode ser que um dos subconjuntos produzidos não seja dividido, enquanto o outro subconjunto é dividido (esse mesmo raciocínio se aplica aos subconjuntos produzidos a partir dos subconjuntos) (Sutton, 2005).

2.5.1.2 Overview de como o algoritmo CART cria uma árvore

CART (*Classification And Regression Trees*) é um método para ajustar árvores para os dados, apresentado em Breiman et al. (2017), e por conta da inviabilidade computacional de escolher a melhor partição, esse método cresce uma larga árvore e então poda a árvore para um tamanho que tem a menor estimação do erro *cross-validation*. O objetivo primário do CART é encontrar partições (divisões) das variáveis de entrada que produzem mínima variância dos valores de resposta (isto é, erro mínimo da soma dos quadrados com respeito à média dos valores de resposta/saída) (Sutton, 2005), (Klusowski, 2019). Primeiramente, o que precisa ser determinado é como dividir subconjuntos de $\mathcal{X}$ (começando pelo próprio $\mathcal{X}$) para produzir um classificador estruturado em árvore. O método CART usa particionamento binário repetitivamente para criar uma árvore binária (outros métodos para criar árvores de classificação permite mais de dois ramos descenderem do mesmo nó, ou seja, um subconjunto de $\mathcal{X}$ pode ser dividido em mais de dois subconjuntos em uma única etapa, mas deve ser comentado, entretanto, que CART não é inferior aos outros métodos por permitir apenas árvores binárias, desde que a partição de $\mathcal{X}$ criada por qualquer outro tipo de estrutura em árvore pode também ser criada usando uma árvore binária) (Sutton, 2005). As dificuldade do método CART incluem:

(1) como fazer cada divisão (identificar qual variável ou quais variáveis deveriam ser utilizada para criar a divisão e determinar o princípio para cada divisão);

(2) como determinar quando um nó da árvore é um nó terminal (um seja, quando um subconjunto não será mais dividido) e

(3) como determinar uma classe prevista para cada nó terminal.

A determinação das classes previstas para os nós terminais é relativamente simples, assim como determinar como serão feitas as divisões, enquanto determinar a árvore de tamanho correto não é uma conclusão tão fácil (Sutton, 2005).

2.5.1.3 Divisão dos nós de uma árvore de decisão

A construção de uma árvore de decisão normalmente é realizada com o objetivo de encontrar um modelo que particione o conjunto de treinamento $\mathcal{L}$ da melhor forma possível. Entre todas as árvores de decisão $\varphi \in \mathcal{H}$, entretanto, pode existir várias árvores que explicam $\mathcal{L}$ igualmente bem. Seguindo o princípio "Navalha de Occam", que tem como base optar pela explicação que faz a menor quantidade de suposições possíveis, ou seja, de escolher a solução mais simples possível que se ajusta aos dados, construir uma árvore de decisão a partir de $\mathcal{L}$ é normalmente reformulado como encontrar a menor árvore φ^* (em termos de nós internos). Enquanto essa suposição faz sentido a partir de um ponto de vista de generalização, também

faz sentido pensando na interpretabilidade, de forma que uma árvore de decisão pequena é mais fácil de entender do que uma árvore larga e complexa (Louppe, 2014).

Define-se uma medida de impureza $\phi(t)$ como uma função que avalia a qualidade de qualquer nó t. Analisando-se o caso de classificação, por exemplo, considerando um nó terminal A, a impureza do nó A é uma função da probabilidade que $y = 1$, escrito como $p(y = 1|A)$, e se A é um nó terminal, idealmente este nó deveria ser composto de casos que são todos iguais a 1 ou todos iguais a 0 (Louppe, 2014),(Berk et al., 2008). Define-se a impureza do nó A então na Equação (1):

$$\phi(A) = \phi[p(y = 1|A)], \tag{1}$$

com $\phi \geq 0$, $\phi(p) = \phi(1-p)$, e $\phi(0) = \phi(1) < \phi(p)$. Ou seja, impureza é não negativa, simétrica com um mínimo quando o nó A contém apenas 0s ou apenas 1s, e possui um máximo quando o nó A contém metade dos casos sendo 0s e metade sendo 1s. A definição de ϕ pode ser feita utilizando uma das seguintes equações, por exemplo: erro de Bayes, função cross-entropia e o índice de Gini, eles são, respectivamente, dados pelas Equações 2, 3 e 4 (Berk et al., 2008), (Cousins; Riondato, 2019):

$$\phi(p) = min(p, 1-p), \tag{2}$$

$$\phi(p) = -p \cdot ln(p) - (1-p) \cdot ln(1-p), \tag{3}$$

e

$$\phi(p) = p(1-p). \tag{4}$$

Sendo essas três funções côncavas, tendo mínimos em $p = 0$ e $p = 1$, e um máximo em $p = 0.5$. As funções cross-entropia e o índice de Gini são as mais utilizadas e geralmente dão resultados muito similares, exceto quando há mais de duas categorias possíveis. Com o objetivo de ilustrar e clarificar o conceito de impureza: para qualquer nó interno, ter-se-á um potencial nó "filho" esquerdo A_L e um potencial "filho" direito A_R. Deseja-se avaliar a utilidade de um potencial particionamento dos dados, a Tabela 1 fornece a informação necessária. Neste exemplo será utilizado a função cross-entropia para a forma de representar a "impureza". Define-se então $y = 1$ se há um sucesso e 0 caso contrário. A estimativa de $p(y = 1|A_L)$ é dada por $n_{12}/n_{1.}$, enquanto a estimativa $p(y = 1|A_R)$ é dada por $n_{22}/n_{2.}$ (Berk et al., 2008).

Segue então que a "impureza" para o "filho da esquerda" é

$$I(A_L) = -\frac{n_{11}}{n_{1.}} ln\left(\frac{n_{11}}{n_{1.}}\right) - \frac{n_{12}}{n_{1.}} ln\left(\frac{n_{12}}{n_{1.}}\right) \tag{5}$$

E a "impureza" para o "filho da direita" é

$$I(A_R) = -\frac{n_{21}}{n_{2.}} ln\left(\frac{n_{21}}{n_{2.}}\right) - \frac{n_{22}}{n_{2.}} ln\left(\frac{n_{22}}{n_{2.}}\right) \tag{6}$$

Tabela 1 – Informação usada para determinar o potencial de uma divisão

	Falha	Sucesso	Total
Nó esquerdo: $x \leq c$	n_{11}	n_{12}	$n_{1.}$
Nó direito: $x > c$	n_{21}	n_{22}	$n_{2.}$
	$n_{.1}$	$n_{.2}$	$n_{..}$

Supondo que no "filho da esquerda" há 400 observações, sendo 300 sucessos e 100 falhas, a impureza é $-0,25(-1,3862) - 0,75(-0,2876) = 0,3465 + 0,2157 = 0,5622$, e que no "filho da direita" há 150 observações, sendo 70 sucessos e 80 falhas, a impureza é $-0,5333(-0,6286) - 0,4666(-0,7621) = 0,3352 + 0,3555 = 0,6907$. Contextualizando esses valores, o maior valor de impureza que poderia ser obtido seria para o caso de 50% de sucessos e 50% de falhas, de forma que o valor (máximo) da impureza seria 0,693, e o mínimo seria 0 para 100% de sucessos. Uma vez que todas as divisões possíveis em cada uma das variáveis possíveis são avaliadas desta forma, uma decisão é feita sobre qual divisão utilizar. O impacto de uma divisão não é apenas uma função da impureza do nó, a importância de cada nó também deve ser levada em conta (Berk et al., 2008), (Cousins; Riondato, 2019).

Define-se então a melhoria resultante de uma divisão como a impureza do "nó pai" menos as impurezas ponderadas da esquerda e da direita. Se este é um número largo, a "impureza" é reduzida substancialmente. Mais formalmente, o benefício da divisão s para o nó A é

$$\Delta I(s,A) = I(A) - p(A_L)I(A_L) - p(A_R)I(A_R), \tag{7}$$

onde $I(A)$ é o valor da impureza do nó "pai", $p(A_R)$ é a probabilidade de um caso caindo no nó "filho" da direita, $p(A_L)$ é a probabilidade de um caso caindo no nó "filho" da esquerda, as outras variáveis são definidas como antes, e $\Delta I(s,A)$ é a variação da impureza. O algoritmo CART encontra o melhor $\Delta I(s,A)$ para cada variável, de forma que a variável e divisão com o maior valor são então escolhidos para definir a nova partição, esse procedimento é utilizado também para todos os nós subsequentes (Berk et al., 2008).

O algoritmo CART consegue continuar particionando até haver apenas uma classe em cada nó, resultando em impureza nula. Diz-se que uma árvore desse tipo é "saturada". Entretanto, bem antes que uma árvore sature, haverão tantos nós terminais para interpretar e o número de casos em cada nó será pequeno. Os nós extremamente pequenos levarão a resultados instáveis, com pequenas mudanças no conjunto de dados levando árvores com estruturas e interpretações diferentes, por exemplo. Uma opção é proibir o algoritmo CART de construir qualquer nó terminal com quantidade de amostras menores que algum valor determinado (Berk et al., 2008).

Começando pelo nó raiz, o algoritmo CART avalia todas as divisões possíveis (utilizando o valor $\Delta I(s,A)$) com cada uma das variáveis de entrada (se a variável independente é idade, por exemplo, e os valores representativos vão de 21 a 24, todas as divisões possíveis mantendo a ordem seriam: 21 versus 22-24, 21-22 versus 23-24 e 21-23 versus 24) utilizando a "impureza" e seleciona a "melhor" divisão geral. Se a variável X tem n valores observados diferentes, há

$(n-1)$ divisões possíveis em X, por outro lado, se X é uma variável categórica tendo m valores observados distintos, há $(2^{m-1}-1)$ divisões possíveis. Os dados são então particionados de acordo com aquela melhor divisão e em seguida o mesmo processo é aplicado para todos os nós subsequentes, até todos os casos terem sido alocados em um nó terminal. Por conta das partições finais não se sobreporem, cada caso aparece em apenas um nó terminal. Assume-se também que quanto menor $i(t)$, mais puro será o nó e melhores serão as predições $\hat{y}_t(\mathbf{x})$ para todo $\mathbf{x} \in \mathcal{L}_t$, onde $\mathcal{L}_t$ é o subconjunto de amostras de treino caindo em t, ou seja, todos ($\mathbf{x}$,y) $\in \mathcal{L}$ tal que $\mathbf{x} \in \mathcal{X}_t$. Começando a partir de um único nó representando o conjunto de treino completo $\mathcal{L}$, árvores de decisão quase ótimas podem então crescer gananciosamente dividindo os nós iterativamente em nós mais puros, ou seja, dividir iterativamente os subconjuntos de $\mathcal{L}$ em subconjuntos menores representados pelos nós, até todos os nós terminais não terem possibilidade de serem mais puros, garantindo predições quase ótimas em $\mathcal{L}$. A suposição gananciosa de tornar a árvore resultante tão pequena quanto possível é feita dividindo cada nó t usando a divisão ótima s^* que maximiza localmente a diminuição de impureza dos nós "filhos"resultantes (Louppe, 2014), (Berk et al., 2008), (Loh, 2014).

Formalmente, a diminuição de impureza de uma divisão binária s é definida como segue:

- **Definição 5**: A diminuição de impureza de uma divisão binária $s \in Q$ dividindo nó t em um nó esquerdo t_L e um nó direito t_R é dado pela Equação 8

$$\Delta i(s,t) = i(t) - p_L i(t_L) - p_R i(t_R), \tag{8}$$

onde p_L (respectivamente, p_R) é a proporção $\frac{N_{t_L}}{N_t}$ (resp., $\frac{N_{t_R}}{N_t}$) das amostras de treino de $\mathcal{L}_t$ indo para t_L (resp., para t_R) e onde N_t é o tamanho do subconjunto $\mathcal{L}_t$ (Louppe, 2014).

2.5.1.4 Classe prevista em um nó terminal de uma árvore de decisão

Considerando-se que o nó t foi declarado como nó terminal de acordo com algum critério de parada. O passo seguinte no procedimento de indução é etiquetar t com um valor constante $\hat{y}_t$ para ser usado como uma predição da variável de saída Y. Como tal, nó t pode ser pensado como um modelo simplista localmente, de forma que produza o mesmo valor de saída $\hat{y}_t$ para todos os valores de entrada possíveis caindo em t. Primeiramente deve-se notar que, para uma árvore φ de estrutura fixada, minimizar o erro de generalização global é equivalente a minimizar o erro de generalização local de cada modelo simplista nos nós terminais. Equacionando,

$$Err(\varphi) = \mathbb{E}_{X,Y}\{L(Y,\varphi(X))\} = \sum_{t \in \tilde{\varphi}} P(X \in \mathcal{X}_t)\mathbb{E}_{X,Y|t}\{L(Y,\hat{y}_t)\}, \tag{9}$$

onde $\tilde{\varphi}$ denota o conjunto de nós terminais da árvore φ e onde a média interna é o erro de generalização local do modelo no nó t. O método de encontrar a melhor árvore de decisão possível (de estrutura fixada) pode ser reformulado então como encontrar as melhores constantes $\hat{y}_t$ em cada nó terminal (Louppe, 2014).

2.5.1.4.1 Classificação

- **Definição 6**: Seja $C(E)$ a classe do exemplo E e seja $C_X(E)$ a classe do exemplo E determinada pelo classificador X. A perda zero-um de X em E, denotada por $L_X(E)$ (ou simplesmente L), é definida pela Equação (10) (Domingos; Pazzani, 1997):

$$L_X(E) = \begin{cases} 0 & \text{se } C_X(E) = C(E) \\ 1 & \text{caso contrário.} \end{cases} \tag{10}$$

Quando L na Equação (9) é a perda zero-um, a média interna é minimizada pela regra da pluralidade (*plurality rule*):

$$\hat{y}_t^* = \arg\min_{c \in y} \mathbb{E}_{X,Y|t}\{1(Y,c)\} \tag{11}$$

$$= \arg\min_{c \in y} P(Y \neq c \mid X \in \mathcal{X}_t) \tag{12}$$

$$= \arg\max_{c \in y} P(Y = c \mid X \in \mathcal{X}_t) \tag{13}$$

Na Equação (11), a expressão $\{1(Y,c)\}$ tem resultado igual a 1 se a classe verdadeira Y for diferente de c e 0 se for igual (reveja a Equação (10) para compreender esta interpretação, se necessário), a expressão $\mathbb{E}_{X,Y|t}$ calcula a média do erro para as amostras que chegam ao nó t, de forma que a equação completa retorna a classe $\hat{y}_t^*$ que no nó t tem o menor erro médio, enquanto que as Equações (12) e (13) têm o mesmo resultado, mas analisando de um ponto de vista de probabilidade e não de erro. Em outras palavras, o erro de generalização de t é minimizado ao prever a classe que é mais provável para as amostras no subespaço de t. É importante notar que se o máximo é obtido por duas ou mais classes diferentes, então $\hat{y}_t^*$ é escolhida arbitrariamente dentre qualquer uma das classes maximizadas (Louppe, 2014).

A Equação (13) não pode ser resolvida sem a distribuição de probabilidade *P(X,Y)*. Entretanto, sua solução pode ser aproximada usando estimativas do erro de generalização local. Seja N_t o número de objetos em $\mathcal{L}_t$ e seja N_{ct} o número de objetos de classe c em $\mathcal{L}_t$. Então, a proporção $\frac{N_{ct}}{N_t}$ pode ser interpretada como a probabilidade estimada $p(Y = c|X \in \mathcal{X}_t)$ (sendo abreviada para $p(c|t)$) da classe c em t e, portanto, sendo utilizada para resolver a Equação (13):

$$\hat{y}_t = \arg\min_{c \in y}(1 - p(c|t)) = \arg\max_{c \in y} p(c|t) \tag{14}$$

De forma similar, define-se a proporção $\frac{N_t}{N}$ como a probabilidade estimada $p(X \in \mathcal{X}_t)$ (abreviada para $p(t)$). Juntando essa estimativa na Equação (9) e aproximando o erro de generalização local com $1 - p(\hat{y}_t|t)$ como feito acima, resulta em:

$$\begin{aligned}
\widehat{Err}(\varphi) &= \sum_{t \in \hat{\varphi}} p(t)(1 - p(\hat{y}_t|t)) \\
&= \sum_{t \in \hat{\varphi}} \frac{N_t}{N} \left(1 - \frac{N_{\hat{y}_t}}{N_t}\right) \\
&= \frac{1}{N} \sum_{t \in \hat{\varphi}} (N_t - N_{\hat{y}_t}) \\
&= \frac{1}{N} \sum_{t \in \hat{\varphi}} \sum_{(\mathbf{x},\, \mathbf{y}) \in \mathcal{L}_t} \mathbf{1}(y \neq \hat{y}_t) \\
&= \frac{1}{N} \sum_{(\mathbf{x},\, \mathbf{y}) \in \mathcal{L}} \mathbf{1}(y \neq \varphi(\mathbf{x})) \\
&= \widehat{Err}^{\text{treino}}(\varphi).
\end{aligned} \tag{15}$$

Uma propriedade importante da regra de atribuição mostrada na Equação (15) é que quanto mais se divide um nó terminal de alguma forma, menor será a estimativa de resubstituição $\widehat{Err}^{treino}(\varphi)$.

Proposição 1. Para qualquer divisão não vazia de um nó terminal t $\in \tilde{\varphi}$ em t_L e t_R, resultando em uma nova árvore φ' onde $\hat{y}_{t_L}$ e $\hat{y}_{t_R}$ são estimados com a Equação (14),

$$\widehat{Err}^{treino}(\varphi) \geq \widehat{Err}^{treino}(\varphi') \tag{16}$$

com igualdade se $\hat{y}_t = \hat{y}_{t_L} = \hat{y}_{t_R}$.

2.5.1.4.2 Determinar a classe prevista para um nó terminal

Se o conjunto de treino pode ser visualizado como sendo um conjunto aleatório da mesma população ou distribuição do qual futuros casos para serem classificados irão vir e se todos os tipos de classificação errôneas são consideradas como igualmente ruins (por exemplo, no caso de $J = 2$, classificar um caso c_1 como c_2 tem a mesma penalidade que classificar um caso c_2 como c_1), a atribuição de classe para nós terminais é feita utilizando a regra da pluralidade (*plurality rule*): a atribuição de classe para um nó terminal é a classe tendo o maior número de membros do conjunto de treino correspondendo àquele nó (de forma que se duas ou mais classes estão juntas por terem os maiores números de casos na amostra de treino, com **x** pertencendo ao conjunto correspondendo ao nó, a classe predita pode ser arbitrariamente selecionada dentre essas classes) (Sutton, 2005).

2.5.1.4.3 Custos e erros de classificação

Até aqui, tem-se procedido com CART classificando pela maioria dos votos. Para cada nó terminal, CART conta o número de casos em uma classe, o número de casos na outra classe e classifica todos os casos como a classe que tem maior quantidade no nó. Quando há mais que duas classes, a classificação é pela categoria com maior número de votos. Considerando-se que em um nó terminal tem 100 casos distribuídos em duas classes, sendo 60 desses casos pertencentes à classe "0" e os outros 40 casos à classe "1", o algoritmo CART atribuiria classe "0" a todos os casos desse nó resultando que os 40 casos que na verdade não pertencem à classe "0" seriam falso-positivos, se o custo de falso-positivos é muito alto, os resultados naquele nó podem ser insatisfatórios. Para ilustrar como funciona o custo, considere um caso onde o algoritmo CART faz a classificação de um conjunto de dados resultando em 100 falso-negativos e 10 falso-positivos, ou seja, uma proporção de 10 falso-negativos para cada 1 falso-positivo, de forma que independente dos custos dos falso-negativos e falso-positivos, os falso-positivos serão tratados como se eles fossem 10 vezes mais caros que os falso-negativos (Berk et al., 2008).

Os custos seguem a seguinte lógica: (i) CART deve introduzir custos quando uma decisão de classificação é feita; (ii) Mesmo se o analista de dados nunca considerar custos, eles são incorporados. A não consideração dos custos resultará na determinação dos custos pelo próprio algoritmo de classificação; (iii) A forma que os casos são classificados ou previstos irão variar dependendo dos custos introduzidos; (iv) Os custos que importam são os custos dos erros de classificação (Berk et al., 2008).

2.5.1.4.4 Probabilidades e custos "a priori"

Os parâmetros que podem ser alterados na árvore de classificação estruturada inclui as probabilidades a priori e os custos variáveis de classificações incorretas (Breiman et al., 2017).

O termo "a priori" vem da estatística Bayesiana, na qual "a priori" refere-se às crenças do analista de dados, antes dos dados serem examinados, sobre a densidade de probabilidade ou a distribuição de algum parâmetro. Suponha que há N observações, C classes para a variável de resposta e K nós terminais. Define-se π_i para $i = 1, 2, ...$, como as probabilidades "a priori" de pertencer à classe i. Para o caso binário, i seria igual a 1 ou 2. $L(i, j)$ é a matriz de perdas para classificações incorretas de casos onde a classe é i, mas foi classificada como j. É essa matriz que informa os custos dos erros de classificação. Para um resultado binário, a matriz é 2 por 2, onde o custo para classificação correta é considerada zero. As probabilidades a priori são um conjunto útil de parâmetros, uma seleção inteligente e ajuste delas podem ajudar em construir uma árvore de classificação desejável (Berk et al., 2008), (Breiman et al., 2017).

Em alguns estudos, os dados podem ser extremamente desbalanceados entre classes. Por exemplo, em uma base de dados de espectros de massa, os compostos sem cloro superavam em quantidade os compostos com cloro por uma razão de 10 para 1. Se as probabilidades a priori são tomadas proporcionalmente à ocorrência de compostos na base de dados, então inicia-se

com uma taxa de classificações incorretas de 10%. Crescer uma árvore de classificação usando tais probabilidades a priori diminui a taxa de classificações incorretas para aproximadamente 5%. Mas o resultado é que compostos sem cloro obtêm uma taxa de classificações incorretas de 3%, enquanto compostos com cloro obtêm uma taxa de classificações incorretas de 30%. O mecanismo produzindo essa disparidade é que se números iguais de compostos com cloro e sem cloro são classificados incorretamente, o efeito na taxa de classificação dos compostos com cloro será muito maior que na taxa dos materiais sem cloro (Breiman et al., 2017).

As probabilidades a priori podem ser utilizadas para ajustar as taxas de erros de classificações de classes individuais em qualquer direção desejada. Por exemplo, tomar probabilidades a priori iguais tende a equalizar as taxas de classificações incorretas. No exemplo dos compostos com cloro, probabilidades a priori iguais resultaram em 9% de classificações incorretas para compostos com cloro e 7% para compostos sem cloro. Colocando uma larga probabilidade a priori em uma classe tenderá a diminuir a taxa de classificações incorretas deles e vice-versa (Breiman et al., 2017).

Em geral, se os custos variáveis de classificações incorretas $C(i|j)$ são especificados, surge então a questão de como incorporar esses custos na regra de divisão. Para o índice de Gini há uma extensão simples. Novamente, considere a regra de classificação subótima na qual, em um nó t, atribui um objeto desconhecido como classe j com probabilidade $p(j|t)$. Note que o custo esperado de estimação usando esta regra é

$$\sum_{j,i} C(i|j)p(i|t)p(j|t) \tag{17}$$

Essa expressão é usada como a medida de Gini da impureza do nó $i(t)$ para custos variáveis de classificações incorretas. No problema com duas classes, a Equação (17) reduz-se a

$$(C(2|1)+c(1|2))p(1|t)p(2|t), \tag{18}$$

resultando no mesmo critério de divisão como no caso de custo unitário (Breiman et al., 2017).

Considera-se A algum nó na árvore e $\tau(x)$ ser a classe correta para uma observação x, onde x representa o vetor de valores das variáveis de entrada para aquela observação. Também considera-se $\tau(A)$ a classe atribuída para o nó A se o nó A é um nó terminal. Finalmente, N_i e N_A são o número de observações na amostra que estão na classe i e no nó A, respectivamente, com N_{iA} sendo o número de observações da classe i no nó A. De forma a sustentar as seguintes afirmações (Berk et al., 2008):

- 1. $P(A)$ (ou equivalentemente, $P[x \in A]$) é a probabilidade de casos aparecendo no nó A, que é equivalente a $\sum_{i=1}^{C} \pi_i P[x \in A|\tau(x) = i]$. Podendo ser estimado por $\sum_{i=1}^{C} \pi_i (N_{iA}/N_i)$. Nota-se que as probabilidades "a priori" afetam diretamente nesses cálculos e, como resultado, pode afetar a estrutura da árvore.

- 2. Então, $p(i|A)$ é a probabilidade da classe i dado que um caso está no nó A, ou $P[\tau(x) = i|x \in A]$. Isto pode ser estimado pelo número de casos da classe i no nó A, dividido pelo

número total de casos naquele nó. Fica evidente que esta probabilidade é igual a $\pi_i P[x \in A|\tau(x) = i]/P[x \in A]$, o qual pode também ser estimado por $\pi_i(N_{iA}/N_i)/\sum_{i=1}^{C}\pi_i(N_{iA}/N_i)$. As probabilidades "a priori" podem fazer uma real diferença porque a probabilidade de um caso da classe i caindo em A depende, em parte, da probabilidade que um caso é verdadeiramente da classe i.

- 3. $R(A)$ é o "risco" associado ao nó A, onde $R(A) = \sum_{i=1}^{C} p(i|A)L(i,\tau(A))$, e onde $\tau(A)$ é escolhido de forma a minimizar o risco. Em outras palavras, o risco associado com nó A é, para o caso binário, a probabilidade de um caso do tipo "1" caindo naquele nó vezes os custos, mais a probabilidade de um caso do tipo "2" caindo naquele nó vezes o custo que segue. Por conta das probabilidades dependerem das probabilidades "a priori", as probabilidades "a priori" afetam os riscos.

- 4. $R(T)$ é o risco da árvore inteira T, esse risco é dado por $\sum_{j=1}^{K} P(A_j)R(A_j)$, onde A_j são os nós terminais da árvore. Nós estamos agora apenas adicionando o risco total associado com cada nó, ponderando pela probabilidade dos casos caindo naquele nó. Isto pode ser também denominado "custo esperado" da árvore inteira (Berk et al., 2008).

E agora o remate. Se $L(i,j) = 1$ para todo $i \neq j$, e as probabilidades "a priori" π_i são tomadas de forma a serem proporções da classe observada na amostra, então $p(i|A) = N_{iA}/N_A$, e R(T) é a proporção classificada incorretamente. O mesmo se aplica para o $R(A)$, o risco associado com qualquer nó terminal particular. Substituindo $L(i,j) = 1$ por $L(i,j) = m$, onde m é alguma constante, apenas aumenta ou diminui o risco por alguma quantidade arbitrária e não faz diferença para o algoritmo CART (Berk et al., 2008).

Portanto, se permitir aos dados determinar tudo, é o mesmo que (a) fazer os custos de todos os erros de classificação representados na função de perda serem os mesmos, e (b) tomar a distribuição empírica como a distribuição "a priori" apropriada. As partições que seguem dependem destas duas condições. Classificação pela votação majoritária dos casos em cada nó terminal também depende destas condições. Se qualquer uma destas condições é diferente, isto pode facilmente levar a diferentes partições e classificações (Berk et al., 2008).

Se o balanceamento entre falso-negativos e falso-positivos em um problema de classificação é insatisfatório, alterar os custos na matriz de perda não resolverá o problema porque o risco associado a um nó é escalado pelo produto entre as probabilidades "a priori" e as entradas na matriz de perda. Para observar as consequências disso, suponha que existe um $\tilde{\pi}$ e um $\tilde{L}$, tal que

$$\tilde{\pi}_i\tilde{L}(i,j) = \pi_i L(i,j) \tag{19}$$

Então, o risco associado com aquele nó é inalterado, e não importa quais valores sejam definidos para $\tilde{\pi}_i$ e $\tilde{L}(i,j)$ desde que a igualdade seja mantida. Se o lado direito da igualdade for pensado como o peso dado para os erros de classificação para a classe i em um dado nó, e se mais ou menos peso é desejado, pode-se alterar as probabilidades "a priori", os custos ou ambos.

Na prática, o trabalho é menor ao alterar apenas um deles, sendo a escolha de qual alterar dependendo de como o *software* é escrito. No caso binário, se os pesos necessitarem serem alterados ao alterar apenas a distribuição "a priori", usa-se a seguinte equação (Berk et al., 2008)

$$\tilde{\pi_i^*} = \frac{\pi_i L_i^*}{\pi_i i L_i^* + \pi_j L_j^*} \tag{20}$$

O índice *i* receberia um valor para a classe classificada corretamente (por exemplo, 1) e outro valor para a classe classificada incorretamente (por exemplo, 2). Os valores de π_i seriam as probabilidades associadas com a distribuição "a priori" empírica. Os valores de L_i^* seriam os novos custos. Por conta da normalização, tudo que importa na matriz de perdas são os custos relativos, ou seja, que o custo de um tipo de erro de classificação é três vezes o custos de outro tipo de erro de classificação, sem necessitar saber dos custos reais.

Será feito um exemplo para clarificar essa explicação. Supondo-se que para um conjunto de dados escolhido, todos os resultados serão determinados apenas pelos dados. Considere o seguinte caso: a distribuição "a priori" empírica é 0,8 para as classificações corretas e 0,2 para as classificações incorretas. O custo de um falso-negativo ou um falso-positivo é 1,0. Supõe-se então que se quer o custo de um falso-negativo ser duas vezes o custo de um falso-positivo: a razão 1 para 1 seria transformada a 1 para 2, utilizando-se a Equação (38). Para classificações corretas, resultaria em $\pi_1 \cdot 1,0 = 0,8 \cdot 1,0 = 0,8$. Enquanto que para classificações incorretas, resultaria em $\pi_2 \cdot 1,0 = 0,2 \cdot 2,0 = 0,4$. Mas então, é preciso normalizar estes valores de forma que a soma das probabilidades resultem em 1,0. Normalizando π_1^*, o cálculo é feito da seguinte forma: $(4/5) * 1,0/(4/5 * 1,0 + 0,2 * 2,0) = 0,67$. Normalizando π_2^*, calcula-se então $(1/5) * 2,0/((1/5) * 2,0 + (4/5) * 1,0) = 0,4/(0,4 + 0,8) = 0,33$. Então, um custo com razão 1 para 2 entre um falso-positivo e um falso-negativo pode ser obtido usando a distribuição "a priori" 0,67 e 0,33. Não há necessidade de alterar os valores em $L(i, j)$, que ainda possui os elementos de custo $L(i \neq j) = 1$. Os resultados obtidos poderiam ser obtidos também utilizando a Equação (38) com $\pi_1 L_1^* = 0,8 * 1,0$ e $\pi_2 L_2^* = 0,2 * 2,0$ (Berk et al., 2008).

Seria conveniente se procedimentos análogos fossem disponíveis para variáveis categóricas de saída com mais de duas categorias. Entretanto, com mais que duas categorias de resposta, é provável que haja mais informação na matriz de perda do que pode ser propriamente capturada por uma distribuição "a priori". Mais especificamente, para qualquer observação dada, o custo de todas as classificações incorretas deve ser o mesmo para que exista uma distribuição "a priori" que podem apropriadamente representar os custos dos erros de classificação (Berk et al., 2008).

Resumidamente, quando a solução CART é determinada apenas pelo dado, a distribuição "a priori" é empiricamente determinada, e os custos na matriz de perda de todos os erros de classificação são os mesmos. Custos estão sendo atribuídos mesmo se o analista de dados não tem conhecimento deles. Se o balanceamento entre falso-negativos e falso-positivos for insatisfatório, esse balanceamento pode ser alterado. Os custos na matriz perda podem ser diretamente alterados, deixando a distribuição "a priori" ser determinada empiricamente, ou a distribuição "a priori" pode ser alterada deixando os custos padrão intocáveis (Berk et al., 2008).

2.5.1.4.5 Poda da árvore de decisão

Com a discussão de custos, pode-se então retornar para o problema de árvores extremamente complexas e o que pode ser feito. Configurar um tamanho de amostra mínimo para cada nó terminal é uma estratégia. Uma segunda estratégia para restringir o tamanho da árvore é chamado "poda". O processo de poda remove galhos indesejáveis através da combinação de nós que não reduzem heterogeneidade suficientemente para a complexidade extra adicionada. Como o algoritmo CART foi substituído, o processo de poda tornou-se menos evidente. Consequentemente, a discussão de poda será feita brevemente (Berk et al., 2008).

Para uma árvore T, o risco geral é dado por

$$R(T) = \sum_{j=1}^{K} P(A_j)R(A_j) \tag{21}$$

Esta é a soma de todos os nós terminais, de forma que o risco associado com cada nó é multiplicado com a probabilidade de casos caindo naquele nó. Pode parecer que uma estratégia aceitável de poda seria simplesmente minizar a Equação (21). Infelizmente, fazer isso deixaria uma árvore saturada intocada. CART construiria nós terminais suficientes tal que todos seriam homogêneos, mesmo se isso resultasse em um nó para cada observação. Com todos os nós terminais homogêneos, o risco associado com cada nó seria zero. Resultando em nós instáveis, *overfitting* realmente preocupante dos dados e muitos detalhes para interpretar de forma útil (Berk et al., 2008).

Uma penalidade pode então ser introduzida para complexidade, de forma que a análise recai em um equilíbrio entre tendência e variância. Com árvores maiores, haverão menos erros de classificação, implicando menos tendência. Mas árvores maiores terão nós terminais com menos casos em cada nó, o que implica maior instabilidade, e portanto, maior variância. O truque é encontrar um balanço entre essas duas variáveis (Berk et al., 2008).

Para levar em conta a complexidade no algoritmo CART, uma solução muito utilizada é definir uma função objetivo, chamada "custo de complexidade", para a poda que inclui uma penalidade explícita para a complexidade. A penalidade não é baseada no número de parâmetros, como na regressão convencional. Para o algoritmo CART, a penalidade é uma função do número de nós terminais. Mais especificamente, tenta-se minimizar a equação

$$R_\alpha(T) = R(T) + \alpha|\tilde{T}|. \tag{22}$$

R_α tem duas partes, uma parte dos custos totais dos erros de classificação para a árvore como um todo, e uma penalidade para a complexidade. Para a segunda parte, $\alpha \geq 0$ é o parâmetro de complexidade e $\tilde{T}$ é o número de nós terminais na árvore T (Berk et al., 2008).

O valor de α quantifica a penalidade para cada nó terminal adicional. Quanto maior o valor de α, mais pesado é a penalidade para complexidade (ou seja, maior é a árvore). Quando

$\alpha = 0$, não há penalidade e resulta em uma árvore saturada. Então, α é o meio pelo qual o tamanho da árvore pode ser determinada (Berk et al., 2008).

Em Breiman et al. (2017) há a prova que para qualquer valor do parâmetro de complexidade α, há uma única subárvore menor de T que minimiza o custo de complexidade. Portanto, não há duas subárvores de mesmo tamanho com o mesmo custo de complexidade, ou seja, dado α, há uma única solução (Berk et al., 2008).

Em tantas implementações do algoritmo CART há formas que o *software* consegue selecionar um valor aceitável para α ou para parâmetros que têm o mesmo objetivo. Alternativamente, pode-se especificar por tentativa e erro um valor de α que leva a cada um dos nós terminais um número suficiente de casos que podem ser sensivelmente interpretados. A interpretação irá depender tanto do número de nós terminais e os tipos de casos que caem em cada nó, tal que um número significante de diferentes modelos de árvores necessitará ser examinado. O maior risco de examinar um largo número de modelos de árvore é acabar levando ao *overfitting*. *Overfitting* já é um problema preocupante em CART, e utilizando informações de tantas árvores poderia apenas piorar as coisas (Berk et al., 2008).

2.5.1.4.6 Árvores de probabilidade de classe através de Gini e poda das árvores

Em alguns problemas, dado um vetor de medição $\mathbf{x}$, o que é solicitado é uma estimativa da probabilidade que o caso está na classe j, sendo $j = 1,...,J$ (Breiman et al., 2017).

Por exemplo, em uma situação de diagnóstico médico, suponha que o paciente pode ter uma de três doenças D_1, D_2, D_3. Em vez de classificar o paciente como tendo uma das três, e descartar as outras duas, pode ser mais adequado estimar as probabilidades relativas do paciente tendo D_1, D_2, ou D_3 (Breiman et al., 2017). Mais precisamente, suponha que os dados são extraídos da distribuição de probabilidade

$$P(A, j) = P(\mathbf{X} \in A, Y = j). \tag{23}$$

Então, quer-se construir estimativas para as probabilidade

$$P(j|\mathbf{x}) = P(Y = j|\mathbf{X} = \mathbf{x}), j = 1,...,J. \tag{24}$$

Em outras palavras, dado que $\mathbf{x}$ é observado, estima-se a probabilidade que o caso está na classe j, $j = 1,...,J$ (Breiman et al., 2017).

Para esse tipo de problema, em vez de construir regras de classificação, quer-se construir regras do tipo

$$\mathbf{d}(\mathbf{x}) = (d(1|\mathbf{x}),...,d(J|\mathbf{X})) \tag{25}$$

com $d(j|\mathbf{x}) \geq 0, j = 1,...,J$, e

$$\sum_j d(j|\mathbf{x}) = 1, \text{para todo } \mathbf{x} \tag{26}$$

Essas regras são chamadas de estimadores de probabilidade de classe. Obviamente, o melhor estimador para este problema, que será chamado de estimador Bayesiano e denotado por $\mathbf{d}_B$, é

$$\mathbf{d}_B = (P(1|\mathbf{x}), ..., P(J|\mathbf{x})). \tag{27}$$

Uma questão crítica é como medir a precisão de um estimador de probabilidade de classe arbitrária. Utilizando E como a esperança de uma variável aleatória. Adota-se o seguinte (Breiman et al., 2017).

Definição 7. A precisão de um estimador $\mathbf{d}$ de probabilidade de classe é definida pelo valor

$$E[\sum_j (P(j|\mathbf{X}) - d(j|\mathbf{X}))^2]. \tag{28}$$

Entretanto, esse critério introduz um problema estranho, desde que depende da probabilidade desconhecida $P(j|\mathbf{x})$ que está sendo estimada. Felizmente, o problema pode ser contornado colocando em uma configuração diferente que resolve esse problema. Defina novas variáveis z_j, $j = 1, ..., J$ por z_j = 1 se Y = j e z_j = 0 para os outros casos (Breiman et al., 2017).

Então

$$E(z_j|\mathbf{X} = \mathbf{x}) = P(Y = j|\mathbf{X} = \mathbf{x}) = P(j|\mathbf{x}) \tag{29}$$

Toma-se $\mathbf{d}(\mathbf{x}) = (d(1|\mathbf{x}), ..., d(J|\mathbf{x}))$ como o estimador de probabilidade de qualquer classe.

Definição 8. O erro quadrático médio (Mean Square Error - MSE) $R^*(\mathbf{d})$ de $\mathbf{d}$ é definido como

$$E[\sum_k (z_j - d(j|\mathbf{X}))^2]. \tag{30}$$

Portanto, o MSE de $\mathbf{d}$ é simplesmente a soma de seus erros quadráticos médios como um preditor das variáveis z_j, $j = 1, ..., J$ (Breiman et al., 2017).

A identidade principal é

Proposição 1. Para qualquer estimador $\mathbf{d}$ de probabilidade de classe

$$R^*(\mathbf{d}) - R^*(\mathbf{d}_B) = E[\sum_j (P(j|\mathbf{X}) - d(j|\mathbf{X}))^2]. \tag{31}$$

Há duas informações interessantes na Proposição 1. A primeira é que entre todos os estimadores de probabilidade de classe, $\mathbf{d}_B$ tem MSE mínimo. A segunda, e mais importante, é que a precisão de $\mathbf{d}$ como definida pela Equação (28), difere de $R^*(\mathbf{d})$ apenas pelo termo constante $R^*(\mathbf{d}_B)$. Portanto, para comparar a precisão de dois estimadores $\mathbf{d}_1$ e $\mathbf{d}_2$, pode-se comparar os valores de $R^*(\mathbf{d}_1)$ e $R^*(\mathbf{d}_2)$. A vantagem significativa obtida aqui é que $R^*(\mathbf{d})$ pode ser estimado utilizando os dados, enquanto precisões não podem (Breiman et al., 2017).

Assume-se que uma árvore T cresceu utilizando uma amostra de aprendizagem ($\mathbf{x}_n$, j_n), $n = 1,...,N$, usando uma regra de divisão não especificada e tem o conjunto de nós terminais $\tilde{T}$. Associado com cada nó terminal t estão as estimativas de resubstituição $p(j|t)$, $j = 1,...,J$, para a probabilidade condicional de estar na classe j dado o nó t (Breiman et al., 2017).

A forma mais simples de usar T como um estimador de probabilidade de classe é definindo: Se $\mathbf{x} \in t$, então

$$\mathbf{d}(\mathbf{x}) = (p(1|t),...,p(J|t)). \tag{32}$$

De forma que $\mathbf{d}$ ou T será usada para denotar esse estimador, dependendo de qual for mais apropriado.

Para cada caso ($\mathbf{x}_n$, j_n) na amostra de aprendizado, defina J valores $z_{n,i}$ por

$$z_{n,i} = \begin{cases} 1 & \text{se } j_n = i, \\ 0 & \text{para os outros casos.} \end{cases}$$

Então, a estimativa de resubstituição $R(T)$ de $R^*(T)$ pode ser formada pelo seguinte raciocínio: Para todo $(\mathbf{x}_n, j_n)$ com $\mathbf{x}_n \in t$, $j_n = j$,

$$\sum_i (z_{n,i} - d(i|\mathbf{x}_n))^2 = (1 - p(j|t))^2 + \sum_{i \neq j} p^2(i|t) = 1 - 2p(j|t) + S \tag{33}$$

onde

$$S = \sum_i p^2(i|t). \tag{34}$$

Então,

$$R(\mathbf{d}) = \sum_{t \in \tilde{T}} \sum_j (1 - 2p(j|t) + S)p(j,t) = \sum_{t \in \tilde{T}} \sum_j (1 - 2p(j|t) + S)p(j|t)p(t) \tag{35}$$

Avaliando a soma sobre j na última expressão resulta em

$$R(\mathbf{d}) = \sum_{t \in \tilde{T}} (1 - S)p(t) \tag{36}$$

O interessante é que

$$1 - S = 1 - \sum_j p^2(j|t) \tag{37}$$

é exatamente o índice de diversidade Gini. Crescer uma árvore usando a regra de divisão Gini continuamente minimiza a estimativa de resubstituição $R(T)$ para o MSE. Em consequência, o uso da regra de divisão Gini é adotada como a melhor estratégia para crescer uma árvore de probabilidade de classe (Breiman et al., 2017).

2.5.1.4.7 Variância do modelo

O algoritmo CART particiona o dado em mais e mais subconjuntos homogêneos, e então atribui uma classe para cada nó terminal. A classe que o CART atribui para cada caso depende do nó terminal para o qual aquele caso é atribuído e a classe atribuída para aquele nó específico. Há várias formas nas quais estes passos podem levar a resultados instáveis. A causa mais óbvia de instabilidade é um pequeno número de observações em qualquer nó. Então, algumas poucas observações podem enviar CART por um tipo de estrutura de ramificação em vez de outro. Pode haver um efeito de inclinação, no qual uma divisão relativamente alta na estrutura da árvore que depende de alguns pontos de dados tem impactos em cascata nas divisões subsequentes. Se aquelas poucas observações são removidas dos dados, ou se elas são substituídas por observações diferentes, uma estrutura de árvore diferente com nós terminais diferentes podem surgir (Berk et al., 2008).

Felizmente, pequenas amostras em nós particulares são facilmente identificados, e há vários bons remédios dentro do *software* CART usual. Pode-se, por exemplo, aumentar o número mínimo de observações em todos os nós ou aumentar o valor da penalidade de complexidade. Tamanhos de nós maiores aparecerão e a estabilidade da saída tenderá a crescer (Berk et al., 2008).

Uma fonte menos aparente de instabilidade deriva dos nós terminais que permanecem relativamente heterogêneos. Se a divisão em um dado nó terminal é próxima do limite de 50-50, o movimento de apenas alguns casos em nós terminais podem mudar as classes atribuídas para cada nó terminal e alterar dramaticamente as classes atribuídas às observações dentro desses nós. Nota-se que isso não é um problema causado por um pequeno número de observações nos nós terminais, entretanto, se houver poucas observações nos nós terminais, a instabilidade causada por proporções quase iguais é agravada (Berk et al., 2008).

Instabilidade resultante devida aos nós terminais heterogêneos é relativamente fácil de identificar quando a distribuição empírica é usada para a distribuição a priori e quando os custos de falso-negativos e falso-positivos são escolhidos para serem os mesmos. Para cada nó terminal, os números de observações em cada classe de resposta podem ser facilmente inspecionados e comparados. Mas quando as classificações atribuídas para os nós terminais dependem de mais do que as contagens dentro do nó, é necessário verificar mais profundamente a saída do CART para determinar o que está acontecendo. Mesmo se for possível concluir que os nós terminais heterogêneos são problemáticos, há pouco o que se fazer. A causa está nos modelos fracos que são incapazes de particionar os dados de forma que resulte em subconjuntos relativamente homogêneos. E sob tais condições, pode não ser inteligente tomar a estrutura de árvore ou valores ajustados muito seriamente (Berk et al., 2008).

2.5.2 Técnicas de reamostragem

2.5.2.1 Bagging e boosting

Bagging e *boosting* são métodos para melhorar as predições de algoritmos de classificação e regressão, sendo chamados de *perturb and combine* (P&C) por Breiman (1998) pelo motivo de serem geradas múltiplas versões do preditor através da perturbação do conjunto de treino ou perturbação do método de construção do próprio preditor (de forma mais concisa, seria a utilização de conjuntos de treinos "perturbados" para construir diferentes árvores, no caso de árvores de classificação, por exemplo, ou "perturbações" no método de construção de cada árvore utilizando a mesma combinação do conjunto de treino), e em seguida fazer a combinação destas múltiplas versões em um único modelo de classificação ou regressão, sendo possível aplicar esses dois métodos em algoritmos baseados em árvore (em *decision tree* ou *random forest*, por exemplo) ou redes neurais (*Artificial Neural Networks* - ANN). É possível acoplar os métodos *bagging* e *boosting* em conjunto com outros algoritmos como a transformada *wavelet*, por exemplo, como realizado no estudo do Belayneh onde foi feita a previsão de seca (Sutton, 2005), (Breiman, 1998), (Belayneh et al., 2016).

Bagging é uma abreviação de *Bootstrap AGGregatING*, proposto em (Breiman, 1996), e é uma técnica de reamostragem com reposição que é usada para interpretação estatística, sendo o *Bootstrap* utilizado para estimar características estatísticas como tendência, variância, intervalos de confiança e funções distribuição, podendo ser então uma excelente técnica para se utilizar com técnicas de ML para encontrar o melhor modelo entre as várias opções disponíveis (Belayneh et al., 2016), de forma mais simples: a mais simples implementação da ideia de gerar conjuntos de treino quase replicados é chamada de *bagging*. A implementação do método *bagging* é feita da seguinte forma: (I) Defina a probabilidade da n-ésima linha no conjunto de treino (sendo o conjunto de treino denotado por T, com tamanho N, ou seja, T consiste de dados $\{(y_n, \mathbf{x}_n), n = 1, ..., N\}$, onde os y's denotam classes diferentes ou uma resposta numérica) ser $p(n) = 1/N$; (II) Para b etapas *bootstrap*: Amostre N vezes da distribuição $p(n)$ (sendo cada vez representada por uma instância, um valor ou uma linha, por exemplo). (I e II - método alternativo) Ou, equivalentemente: Faça amostragens de T com substituição, formando um conjunto de treino reamostrado T' (T' é chamado de amostra/conjunto *bootstrap* de T, T' terá tamanho N, mesmo tamanho do conjunto de dados original), de forma que certos exemplos podem aparecer mais de uma vez em um conjunto *bootstrap*, enquanto outros exemplos podem não aparecer. Em média, 63,2% das instâncias (linhas) são selecionadas e as 36,8% das restantes não são selecionadas. As instâncias selecionadas são designadas como conjunto de treino, enquanto as linhas não escolhidas se tornam o conjunto de teste. (III) Denote a distribuição em T dada por $p(n)$ como $P^{(B)}$. Então T' é individual e identicamente distribuído (iid) de P^B; (IV) Repita esse procedimento de amostragem, obtendo uma sequência de conjuntos de treino *bootstrap* independentes; (V) Construa uma sequência de modelos de classificação usando o mesmo algoritmo de classificação aplicado para cada um dos b conjuntos de treino *bootstrap*; (VI) Então deixe os múltiplos

modelos de classificação votarem para cada classe (se for um algoritmo de classificação) ou faça uma média dos resultados dos modelos (se for um algoritmo de regressão). Para qualquer ponto (linha/instância) $\mathbf{x}$, $C_A(\mathbf{x})$ (onde C_A representa o modelo de classificação agregado, sendo o termo agregado utilizado para representar a combinação dos modelos a fim de obter o melhor resultado (Kittler; Roli, 2000)) depende da probabilidade P que os conjuntos de treino possuem de $C_A(\mathbf{x}) = C_A(\mathbf{x}, P)$. Resultando no modelo de classificação *bagged* como $C_A(\mathbf{x}, P^{(B)})$ (Breiman, 1998), (Raschka, 2018), (Nakatsu, 2023), (Ghojogh; Crowley, 2019), (Breiman, 1996).

Boosting foi originalmente desenvolvido para classificação sendo normalmente aplicado para *weak learners* (podendo ser aplicado também para modelos precisos, como árvores cuidadosamente crescidas e podadas, servindo como *base learner*). Para um modelo de classificação de duas classes, um *weak learner* é um algoritmo de classificação que pode ser apenas um pouco melhor que adivinhação aleatória (como adivinhação aleatória tem uma taxa de erro de 0,5, um *weak classifier* apenas tem que prever corretamente, em média, um pouco mais que 50% das vezes) (Sutton, 2005).

Boosting, da mesma forma que *bagging*, é um método que utiliza votação para classificação e regressão, melhorando a precisão de ambos os tipos de modelos. Diferente de *bagging* que usa uma simples média dos resultados para obter uma predição geral, *boosting* utiliza a combinação de vários modelos, onde cada um iterativamente criado através de versões ponderadas da amostra de aprendizado com os pesos ajustados a cada passo para aumentar o peso dos casos previstos incorretamente no passo anterior e no final usa uma média ponderada dos resultados obtidos destes vários criados, ou seja, a ideia principal do *boosting* é produzir uma sequência de modelos, de forma que cada modelo subsequente concentra-se em treinar casos que não foram bem previstos pelo modelo anterior (Sutton, 2005), (Belayneh et al., 2016). O algoritmo *boosting* mais popular é chamado de "AdaBoost.M1"(também conhecido como "*Discrete AdaBoost*"), apresentado neste trabalho como Algoritmo 1 (Freund; Schapire, 1997). Considere um problema de classificação de duas classes, com a variável dependente (variável de saída) dada por $Y \in \{-1, 1\}$ e as variáveis independentes representadas por um vetor X, um modelo de classificação $G(X)$ produz uma predição possuindo um dos valores {-1,1}. Sendo a taxa de erro no conjunto de treino dada pela Equação (38):

$$\overline{err} = \frac{1}{N}\sum_{i=1}^{N} I(y_i \neq G(x_i)), \tag{38}$$

onde I é a função indicadora que retorna 1 se a condição $(y_i \neq G(x_i)$ for verdadeira e 0 se a condição for falsa (Friedman, 2001). A metodologia do *boosting*, mais detalhadamente, é aplicar sequencialmente o algoritmo *weak learner* para versões modificadas dos dados, produzindo então uma sequência de modelos fracos $G_m(x), m = 1, 2, \ldots, M$. Onde essas versões modificadas dos dados em cada etapa do *boosting* consiste de aplicar pesos $w_1, w_2, \ldots, w_n$ para cada observação (instância) do conjunto de treino (x_i, y_i), i = 1,2,...,N. Inicialmente, todos os pesos possuem valor $w_i = 1/N$, de forma que a primeira etapa simplesmente treina o algoritmo de classificação no dado normalmente. Para cada iteração sucessiva (m = 2,3,..,M) os pesos de cada observação são

alterados e o algoritmo de classificação é novamente aplicado para as observações ponderadas. Na etapa m, as instâncias que foram categorizadas incorretamente pelo algoritmo de classificação do passo anterior $G_{m-1}(x)$ têm seus pesos aumentados, enquanto as que foram categorizadas corretamente têm seus pesos reduzidos. De forma que enquanto as iterações continuam, observações que são difíceis de serem categorizadas corretamente recebem um peso cada vez maior, e o algoritmo de classificação é então forçado a se concentrar nessas observações incorretamente categorizadas nos passos anteriores (Hastie et al., 2009). Com *"sign"* representando a função sinal, ou seja, ela retornará a classificação prevista pela maioria dos modelos treinados para cada instância (onde a votação dos modelos são representados pelo somatório dentro dos colchetes).

Algoritmo 1: AdaBoost.M1

1. Inicializa os pesos de cada instância $w_i = \frac{1}{N}$, $i = 1, 2, \ldots, N$.
2. Para $m = 1$ a M:
   (a) Treina o algoritmo de classificação $G_m(x)$ com o conjunto de treino usando os pesos w_i.
   (b) Calcule a taxa de erro de cada modelo fraco

$$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$

   (c) Calcule $\alpha_m = \log\left(\frac{1-\text{err}_m}{\text{err}_m}\right)$.
   (d) Defina $w_i \leftarrow w_i \cdot \exp\left[\alpha_m \cdot I(y_i \neq G_m(x_i))\right]$, $i = 1, 2, \ldots, N$.
3. Resultando em $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

Alguns estudos indicam que *boosting* é um dos métodos mais úteis no aprendizado estatístico e/ou ML, enquanto outros também indicam que a aplicação de *boosting* em árvores de classificação resulta em modelos de classificação que geralmente são competitivos com qualquer outro algoritmo de classificação (Sutton, 2005).

2.5.2.2 Validação cruzada

Validação cruzada (*Cross-validation* - CV) ou *K-fold cross-validation* (KCV) é uma técnica de reamostragem baseada na divisão dos dados para fazer avaliação preditiva dos modelos estatísticos (Bates; Hastie; Tibshirani, 2024), (Yates et al., 2023). Mais especificamente, CV funciona através da divisão dos dados disponíveis em um par de conjuntos de treino e teste, onde o modelo é ajustado para os dados de treino e após isso, utiliza-se esse modelo para prever as saídas do conjunto de teste, através da repetição desse processo para tantas divisões diferentes dos dados, e a performance preditiva média de um ou mais modelos é estimada (Yates et al., 2023). A técnica KCV permite tanto ajustar os hiperparâmetros (grau de suavidade de uma spline, por exemplo (Yates et al., 2023)) quanto estimar o erro de generalização (Anguita et al., 2012). No método KCV, o conjunto de dados disponível é dividido em k *folds*/divisões (o valor de k é escolhido e fixo aprioristicamente, normalmente sendo igual a 5, 10 ou 20 (Anguita

et al., 2012). Algumas considerações importantes em valores para k são feitas em Yates et al. (2023)), e iterativamente o modelo é treinado em todas as divisões, exceto uma, que é utilizada para a avaliação. Então a avaliação acontece em uma parte dos dados que não foi utilizada para o treino, de forma que *overfitting* não influencia a avaliação. Esse procedimento é repetido k vezes (alternando os *folds* utilizados para treino e teste) e uma média pode então ser calculada (Anguita et al., 2012), (Probst, 2019). O principal objetivo de usar validação cruzada é para encontrar um modelo que leva à menor discrepância geral possível para um tamanho de amostra fornecido, não para encontrar um modelo com erro de aproximação nulo (Browne, 2000).

2.5.2.3 Funções basis

Em problemas de regressão, $f(X) = E(Y|X)$ (onde X e Y representam as variáveis ou o conjunto de variáveis, independentes e dependentes, respectivamente) irá tipicamente ser não linear e não aditiva em X, e representando $f(X)$ por um modelo linear é geralmente uma aproximação conveniente e algumas vezes necessária. A ideia principal por trás de uma função *"basis"* é aumentar/substituir o vetor de entradas X com variáveis adicionais, que são transformações de X, e então usar modelos lineares neste novo espaço de características de entrada derivadas (Hastie et al., 2009). Denota-se por $b(X) : \mathbb{R}^p \mapsto \mathbb{R}$ a m-ésima transformação de X, $m = 1, \ldots, M$. E então modela-se a Equação (39)

$$f(X) = \sum_{m=1}^{M} \beta_m b(X), \tag{39}$$

uma expansão linear *"basis"* em X. A utilidade desse método é que uma vez que as funções *"basis"* b tenham sido determinadas, os modelos serão lineares nestas novas variáveis. Nesse trabalho, as funções *"basis"* são os modelos de classificação individuais $G_m(x) \in \{-1, 1\}$. *Boosting* é uma das formas de ajustar uma expansão aditiva em um conjunto de funções *"basis"* elementares. Mais geralmente, expansões em funções *"basis"* tomam a forma da Equação (40) (Hastie et al., 2009)

$$f(X; \gamma_m) = \sum_{m=1}^{M} \beta_m b(x; \gamma_m), \tag{40}$$

onde $\beta_m, m = 1, 2, ..., M$ são os coeficientes de expansão e $b(x; \gamma_m) \in \mathbb{R}$ normalmente são funções simples do argumento multivariável x, caracterizado por um conjunto de parâmetros γ_m. Para uma árvore de regressão, por exemplo, os parâmetros γ_m são as variáveis de divisão (*splitting*), localizações de cada divisão (*split*) e os significados dos nós terminais das árvores individuais (Hastie et al., 2009), (Friedman, 2001). Em geral, escolher um modelo parametrizado $f(X; P)$ (onde $P = \gamma_m$) muda o problema de uma otimização de função para um de otimização de parâmetro como é mostrado na Equação (41),

$$P^* = \arg\min_P \Phi(P), \tag{41}$$

sabendo que P^* representa o valor ótimo para o parâmetro P, onde

$$\Phi(P) = E_{y,x}L(y, f(x;P)), \tag{42}$$

onde L é a função de perda utilizada, podendo ser o erro quadrático ou erro absoluto, por exemplo. Resultando na Equação (43)

$$F^*(x) = F(x;P^*). \tag{43}$$

A maioria das funções $f(x;P)$ e L devem ser resolvidas através de métodos de otimização numérica como na Equação (41). De forma que isso normalmente envolve expressar a solução para os parâmetros na forma da Equação (44)

$$P^* = \sum_{m=0}^{M} p_m, \tag{44}$$

onde p_0 é uma tentativa inicial e $\{p_m\}_1^M$ são incrementos sucessivos ("passos" ou *"boosts"*), sendo cada um baseados na sequência dos passos anteriores. A forma para computar cada passo p_m é definido pelo método de otimização selecionado.

2.5.3 Floresta aleatória (*Random Forest* - RF)

Random Forest (RF) aqui irá referir-se ao algoritmo original publicado em (Breiman, 2001), tendo a possibilidade de usar este algoritmo tanto em problemas de regressão quanto classificação. O algoritmo RF é versátil o suficiente para ser aplicado em problemas de larga escala e retorna medidas de importância de variável. Um dos motivos de ser um dos algoritmos mais utilizados é pela capacidade de escalabilidade com uma quantidade muito grande de informações, além de poder ser utilizado em conjuntos de dados pequenos e espaços de características com alta dimensionalidade (Biau; , 2016).

No lado teórico, a história de florestas aleatórias é menos conclusiva e, apesar de ser um dos algoritmos mais utilizados atualmente, pouco é conhecido sobre as propriedades matemáticas do método. O resultado teórico com mais destaque é de Breiman (2001), que comenta a respeito do erro de generalização das árvores em termos de correlação e força das árvores individuais. A dificuldade em analisar as florestas aleatórias apropriadamente pode ser explicada pela propriedade de caixa preta (*black-box*) do método, que é com certeza uma combinação sutil de diferentes componentes. Entre os ingredientes essenciais das florestas aleatórias, *bagging* de Breiman (1996) e o critério de divisão CART de Breiman et al. (2017) desempenham papéis fundamentais (Biau; , 2016).

Entretanto, enquanto *bagging* e a divisão CART desempenham papéis-chave no algoritmo RF, ambos são difíceis de analisar com rigor matemático, explicando então por que estudos teóricos consideram versões simplificadas do procedimento original. Isso é feito frequentemente simplesmente ignorando o passo *bagging* e/ou substituindo a seleção de divisão CART por um

método mais simples. Assim, o foco nessa subseção será o algoritmo RF aplicado para regressão. (Biau; , 2016).

A estruturação geral é uma estimação da regressão na qual um vetor aleatório de entrada $\mathbf{X} \in \mathcal{X} \subset \mathbb{R}^p$ é observado e o objetivo é prever a resposta aleatória $Y \in \mathbb{R}$ pela estimação da função regressão $m(\mathbf{x}) = \mathbb{E}[Y|\mathbf{X} = \mathbf{x}]$. Com esse propósito em mente, assume-se que é utilizada uma amostra de treino $\mathcal{D}_n = ((\mathbf{X}_1, Y_1), ..., (\mathbf{X}_n, Y_n))$ de variáveis aleatórias independentemente distribuídas. O objetivo é usar $\mathcal{D}_n$ para construir uma estimativa $m_n : \mathcal{X} \to \mathbb{R}$ da função m. De forma que diz-se que a estimativa da função de regressão m_n é (*Mean Squared Error* - MSE) consistente se $\mathbb{E}[m_n(\mathbf{X}) - m(\mathbf{X})]^2 \to 0$ quando $n \to \infty$ (Biau; , 2016).

O algoritmo RF (para regressão) é então um preditor compondo-se de uma coleção de M árvores de regressão randomizadas, podendo essa randomização ser feita randomizando o conjunto de características da árvore, o conjunto de dados ou as duas formas ao mesmo tempo (Biau; , 2016), (Scornet, 2016). Para a j-ésima árvore na floresta, o valor predito no ponto de investigação $\mathbf{x}$ é representado por $m_n(\mathbf{x}; \Theta_j, \mathcal{D}_n)$, onde $\Theta_1, ..., \Theta_M$ são variáveis aleatórias independentes, distribuídas da mesma forma que uma variável aleatória genérica Θ. Efetivamente, a variável Θ é usada para reamostrar o conjunto de treino anterior para o crescimento das árvores individuais e para selecionar as direções consecutivas de divisão (Biau; , 2016). A representação matemática para a estimação da j-ésima árvore é apresentada na Equação (45)

$$m_n(\mathbf{x}; \Theta_j, \mathcal{D}_n) = \sum_{i \in \mathcal{D}_n^\star(\Theta_j)} \frac{\mathbb{1}_{\mathbf{X}_i \in A_n(\mathbf{x}; \Theta_j, \mathcal{D}_n)} \mathbf{Y}_i}{N_n(\mathbf{x}; \Theta_j, \mathcal{D}_n)}, \tag{45}$$

onde $\mathcal{D}_n^\star(\Theta_j)$ é o conjunto de pontos de dados selecionados anteriormente à construção da árvore, $A_n(\mathbf{x}; \Theta_j, \mathcal{D}_n)$ é a célula/região contendo $\mathbf{x}$, e $N_n(\mathbf{x}; \Theta_j, \mathcal{D}_n)$ é o número de pontos (selecionados) que caem em $A_n(\mathbf{x}; \Theta_j, \mathcal{D}_n)$, de forma que o numerador é $1 \cdot \mathbf{Y}_i$ se X_i cair na mesma região que a instância de teste/previsão $\mathbf{x}$ e 0 para demais casos. Escrevendo de outra forma, como para cada $\mathbf{X}_i$ há um $\mathbf{Y}_i$, se $\mathbf{x}$ cair no nó/célula que contém $\mathbf{X}_i$, um dos termos do somatório terá valor $\mathbf{Y}_i$ dividido pela quantidade de casos N_n que caíram naquele nó A_n (Biau; , 2016). Nesta etapa, pode-se observar que as M árvores são associadas para formar a estimação da floresta aleatória (finita), sendo esta estimativa representada pela Equação (46):

$$m_{M,n}(\mathbf{x}; \Theta_1, \ldots, \Theta_M, \mathcal{D}_n) = \frac{1}{M} \sum_{j=1}^{M} m_n(\mathbf{x}; \Theta_j, \mathcal{D}_n). \tag{46}$$

Desde que a quantidade M de árvores pode ser escolhida arbitrariamente larga (limitada apenas pelos recursos computacionais disponíveis ou tempo que o usuário quer gastar nessa etapa do algoritmo, por exemplo), pode-se deixar M tender ao infinito, resultando na estimação da floresta aleatória (infinita) representada pela Equação (47):

$$m_{\infty,n}(\mathbf{x}; \mathcal{D}_n) = \mathbb{E}_\Theta[m_n(\mathbf{x}; \Theta, \mathcal{D}_n)] \tag{47}$$

onde $\mathbb{E}_{\Theta}$ representa a esperança (média) com respeito ao parâmetro aleatório Θ e condicionada a $\mathcal{D}_n$. A operação "$M \to \inf$" é fundamentada pela lei dos grandes números (*law of large numbers*), que afirma quase seguramente (condicionado a $\mathcal{D}_n$) a Equação (48),

$$\lim_{M\to\infty} m_{M,n}(\mathbf{x}; \Theta_1, \ldots, \Theta_M, \mathcal{D}_n) = m_{\infty,n}(\mathbf{x}; \mathcal{D}_n) \tag{48}$$

Nas próximas etapas, será utilizada a expressão $m_{\infty,n}(\mathbf{x})$ em vez de $m_{\infty,n}(\mathbf{x}; \mathcal{D}_n)$ com o objetivo de simplificar as notações (Biau; , 2016).

2.5.3.1 Algoritmo RF

Nas florestas originais de Breiman (2001), cada nó de uma árvore individual é vinculada com uma célula hiper-retangular. A raiz da árvore é a própria partição $\mathcal{X}$ e a cada passo da construção da árvore, um nó (ou equivalentemente, sua célula correspondente) é dividido em dois caminhos distintos. Os nós terminais (ou folhas), tomados juntos, formam uma partição de $\mathcal{X}$. O algoritmo opera crescendo M árvores diferentes da seguinte forma: (I) Anteriormente à construção de cada árvore, a_n observações são escolhidas aleatoriamente com ou sem reposição a partir do conjunto de dados original. Estas, e apenas estas, a_n observações (com possíveis repetições) são levadas em conta na construção da árvore. De forma que em cada célula de cada árvore há uma divisão realizada pela maximização do critério CART em várias direções *mtry* escolhidas igualmente randomizadas entre os p originais (onde p é a dimensão dos vetores de entrada $X \in \mathcal{X}$ e o subconjunto das coordenadas selecionadas é chamado $\mathcal{M}_{try}$). Por último, a construção das árvores individuais é interrompida quando cada célula/nó terminal contém menos pontos que o tamanho do nó (sendo o tamanho do nó definido pelo parâmetro *min_samples_leaf* na função RandomForestRegressor da biblioteca *scikit-learn*, cada folha contém um pequeno número de observações, normalmente entre 1 e 5) (Biau; , 2016). Para qualquer ponto de consulta $\mathbf{x} \in \mathcal{X}$, cada árvore de regressão prevê a média do valor Y_i para o qual o correspondente $\mathbf{X}_i$ cai na célula de $\mathbf{x}$. Deve-se chamar atenção para o fato que crescer a árvore e fazer a estimação final apenas depende dos pontos de dados a_n previamente selecionados (Biau; , 2016). O Algoritmo 2 descreve completamente como calcular a predição de uma floresta, possuindo três parâmetros importantes:

- 1. $a_n \in \{1, ..., n\}$: o número de pontos de dados amostrados em cada árvore;
- 2. $mtry \in \{1, .., p\}$: o número de direções possíveis para divisão em cada nó da árvore;
- 3. $min\_samples\_leaf \in \{1, ..., a_n\}$: o número de exemplos em cada célula indicando que não haverá mais divisão.

Algoritmo 2: Valor em **x** predito pela floresta aleatória de Breiman

Entrada: Conjunto de treino $\mathcal{D}_n$, número de árvores M > 0, $a_n \in \{1,...,n\}$, *mtry* $\in \{1,...,p\}$, *min_samples_leaf* $\in \{1,...,a_n\}$ e $\mathbf{x} \in \mathcal{X}$.

Saída: Previsão da floresta aleatória em **x**.

1. **Para** $j = 1$ **a** $j = M$, **faça**:

   a) Selecione a_n pontos, com (ou sem) reposição, uniformemente em $\mathcal{D}_n$. Nos passos seguintes, apenas estas observações a_n são utilizadas

   b) Defina $\mathcal{P} \leftarrow (\mathcal{X})$ a lista contendo a célula associada com a raiz da árvore.

   c) Defina $\mathcal{P}_{\text{final}} \leftarrow \emptyset$ uma lista vazia.

   d) **Enquanto** $\mathcal{P} \neq \emptyset$, **faça**:

      i. Deixe A ser o primeiro elemento de $\mathcal{P}$.

      ii. **Se** (A contém menos pontos que o valor de $<$ *min_samples_leaf*) **ou** (se todos os $X_i \in A$ são iguais), **então**:

      - Remova a célula A da lista $\mathcal{P}$.
      - $\mathcal{P}_{\text{final}} \leftarrow \text{Concatenar}(\mathcal{P}_{\text{final}}, A)$.

      iii. **Do contrário**:

      - Selecione uniformemente, sem reposição, um subconjunto $\mathcal{M}_{\text{try}} \subset \{1,\ldots,p\}$ de cardinalidade *mtry*.
      - Selecione a melhor divisão em A otimizando o critério de divisão CART em conjunto com as coordenadas em $\mathcal{M}_{try}$.
      - Corte a célula A de acordo com a melhor divisão. Defina A_L (A-*left* ou A-esquerda) e A_R (A-*right* ou A-direita) as duas células resultantes.
      - $\mathcal{P} \leftarrow \text{Concatenar}(\mathcal{P}, A_L, A_R)$.

   e) Calcule o valor predito $m_n(\mathbf{x}; \Theta_j, \mathcal{D}_n)$ em **x** igual à média dos Y_i que caem na célula de **x** na partição $\mathcal{P}_{final}$.

2. Calcule a estimação $m_{M,n}(\mathbf{x}; \Theta_1, \ldots, \Theta_M, \mathcal{D}_n)$ no ponto de consulta **x** de acordo com a Equação (46).

Ainda deve ser explicado como o critério de divisão CART funciona. Para facilitar a compreensão, considera-se uma árvore sem subamostragem, ou seja, a árvore utiliza o conjunto de dados original $\mathcal{D}_n$ completamente para sua construção. Além do mais, considera-se A uma célula genérica e estabelece N_n(A) o número de pontos de dados caindo em A. Um corte em A é um par (j, z), onde j é algum valor (dimensão) de $\{1,...,p\}$ e z a posição do corte ao longo da j-ésima coordenada, dentro dos limites de A. Deixe $\mathcal{C}_A$ ser o conjunto de todos os possíveis cortes em A. Então, com a notação $\mathbf{X}_i = (\mathbf{X}_i^{(1)}, ..., \mathbf{X}_i^{(p)})$, para qualquer ($j$,$z$) $\in \mathcal{C}_A$, o critério de

divisão CART toma a forma mostrada na Equação (49)

$$L_{reg,n}(j,z) = \frac{1}{N_n(A)} \sum_{i=1}^{n} (Y_i - \bar{Y}_a)^2 \mathbb{1}_{\mathbf{X}_i \in A} - \frac{1}{N_n(A)} \sum_{i=1}^{n} (Y_i - \bar{Y}_{A_L} \mathbb{1}_{\mathbf{X}_i^{(j)} < z} - \bar{Y}_{A_R} \mathbb{1}_{\mathbf{X}_i^{(j)} \geq z})^2 \mathbb{1}_{\mathbf{X}_i \in A}, \quad (49)$$

onde $A_L = \{\mathbf{x} \in A : \mathbf{x}^{(j)} < z\}$, $A_R = \{\mathbf{x} \in A : \mathbf{x}^{(j)} \geq z\}$ e $\bar{Y}_A$ é a média dos Y_i tal que $\mathbf{X}_i$ pertence a A, com a convenção que a média é igual a 0 quando nenhum ponto $\mathbf{X}_i$ pertence a A. Para cada célula A, o corte ótimo $(j_n^\star, z_n^\star)$ é selecionado pela maximização $L_{reg,n}(j,z)$ apresentada na Equação 50 sobre $\mathcal{M}_{try}$ e $\mathcal{C}_A$, ou seja,

$$\{j_n^*, z_n^*\} \in \arg \max_{\substack{j \in \mathcal{M}_{\text{try}} \\ (j,z) \in \mathcal{C}_A}} \mathcal{L}_{\text{reg},n}(j,z) \quad (50)$$

É interessante destacar que a otimização da Equação (50) pode ser utilizada para os casos de reamostragem, otimizando as a_n observações pré-selecionadas em vez do conjunto de dados original $\mathcal{D}_n$.

Portanto, em cada célula de cada árvore, o algoritmo escolhe aleatoriamente *mtry* coordenadas em {1,...,p}, o critério (49) avalia todos os cortes possíveis ao longo das direções $\mathcal{M}_{try}$ e retorna a melhor possibilidade. A medida de qualidade (49) é o critério usado no algoritmo CART de (Breiman et al., 2017), onde esse critério calcula a (renormalizada) diferença entre a variância no nó antes e depois do corte ser feito. Existem diferenças principais entre CART e uma árvore da floresta de (Breiman, 2001). Nas florestas de Breiman (2001), o critério (49) é avaliado em um subconjunto $\mathcal{M}_{try}$ de coordenadas aleatoriamente selecionadas e não em cima da faixa {1,...,p} completa. Além do mais, as árvores individuais não são podadas no algoritmo RF (Breiman, 2001), e as células finais não contém mais observações que o valor de *nodesize* (a menos que todos os pontos de dados na célula tem o mesmo $\mathbf{X}_i$). Por último, cada árvore é construída em um subconjunto de exemplos a_n coletados dentro da amostra inicial, não na amostra por inteiro $\mathcal{D}_n$, e apenas estas observações a_n são usadas para o cálculo da estimação. Quando $a_n = n$ (e a reamostragem é feita com reposição), o algoritmo é executado no modo *bootstrap*, enquanto $a_n < n$ corresponde ao caso de subsamostragem (com ou sem reposição) (Biau; , 2016).

2.5.3.2 Funções Borel

Definição 9. Se g é uma função de valor real de uma única variável real, ela é uma função Borel se a imagem inversa de todo conjunto Borel é um conjunto Borel.

Uma análise teórica mais detalhada mostra que a classe de funções Borel inclui qualquer função contínua e qualquer função que é contínua exceto em pontos discretos onde há "pulos" de descontinuidade. A noção de funções Borel não é limitada a funções de valor real de uma única variável real, as funções podem ser vetoriais, e o domínio pode ser formado por espaços Euclidianos de qualquer dimensão. Um motivo para a importância de funções Borel em teoria de

probabilidades é que uma função Borel de uma variável aleatória é uma nova variável aleatória (Pfeiffer, 2012).

2.5.3.3 Classificação supervisionada

Para simplificar, será considerado apenas o problema de classificação binária, mantendo em mente que florestas aleatórias são intrinsecamente capazes de lidar com problemas multi-classe. Nesta configuração, a resposta aleatória Y toma valores em $\{0,1\}$ e, dado $\mathbf{X}$, faz-se a tentativa de adivinhar o valor Y. Um classificador, ou regra de classificação, m_n é uma função Borel mensurável de $\mathbf{X}$ e $\mathcal{D}_n$ que tenta estimar o a etiqueta Y a partir de $\mathbf{X}$ e $\mathcal{D}_n$. Nesse contexto, pode-se dizer que o classificador m_n é consistente se a probabilidade do erro

$$L(m_n) = \mathbb{P}\big[m_n(\mathbf{X}) \neq Y\big] \xrightarrow[n\to\infty]{} L^\star \tag{51}$$

onde $L^\star$ é o erro do ótimo, mas desconhecido, classificador Bayes:

$$m^\star(\mathbf{x}) = \begin{cases} 1 & \text{se } \mathbb{P}[Y=1 \mid \mathbf{X}=\mathbf{x}] > \mathbb{P}[Y=0 \mid \mathbf{X}=\mathbf{x}], \\ 0 & \text{para os outros casos.} \end{cases} \tag{52}$$

No contexto de classificação, o classificador floresta aleatória é obtido através de maioria dos votos entre as árvores de classificação, ou seja,

$$m_{M,n}(\mathbf{x};\Theta_1,\ldots,\Theta_M,\mathcal{D}_n) = \begin{cases} 1 & \text{se } \frac{1}{M}\sum_{j=1}^{M} m_n(\mathbf{x};\Theta_j,\mathcal{D}_n) \geq 1/2, \\ 0 & \text{para os outros casos.} \end{cases} \tag{53}$$

Se uma folha representa uma região A, então uma árvore aleatória de classificação toma a forma simplificada

$$m_n(\mathbf{x};\Theta_j,\mathcal{D}_n) = \begin{cases} 1 & \text{se } \displaystyle\sum_{i\in\mathcal{D}_n^\star(\Theta_j)} \mathbf{1}_{\{\mathbf{X}_i\in A, Y_i=1\}} > \sum_{i\in\mathcal{D}_n^\star(\Theta_j)} \mathbf{1}_{\{\mathbf{X}_i\in A, Y_i=0\}},\ \mathbf{x}\in A, \\ 0 & \text{para os outros casos.} \end{cases} \tag{54}$$

onde $\mathcal{D}_n^\star(\Theta_j)$ contém os pontos de dados selecionados no passo de reamostragem, isto é, em cada folha, uma votação é realizada sobre $(\mathbf{X}_i, Y_i)$ para o qual $\mathbf{X}_i$ está na mesma região. O algoritmo 2 pode ser facilmente adaptado para o problema de classificação de duas classes sem modificar o critério de divisão CART. Para verificar isso, toma-se $Y \in \{0,1\}$ e considera-se uma única árvore sem subamostragem. Para qualquer célula genérica A, toma-se $p_{0,n}(A)$ (respectivamente, $p_{1,n}(A)$) como a probabilidade empírica, dado um ponto de dado em uma célula A, tem-se a etiqueta igual a 0 (respectivamente, etiqueta igual a 1). Notando-se que $\bar{Y}_A = p_{1,n}(A) = 1 - p_{0,n}(A)$, o

critério de divisão CART para classificação (esse critério é baseado na medida de impureza de Gini $2p_{0,n}(A)p_{1,n}(A)$), para qualquer $(j,z) \in \mathcal{C}_A$ é dado por

$$L_{classe,n}(j,z) = p_{0,n}(A)p_{1,n}(A) - \frac{N_n(A_L)}{N_n(A)} \cdot p_{0,n}(A_L)p_{1,n}(A_L) - \frac{N_n(A_R)}{N_n(A)} \cdot p_{0,n}(A_R)p_{1,n}(A_R) \quad (55)$$

2.5.3.4 Mecanismo de reamostragem no algoritmo floresta aleatória

A etapa de reamostragem no algoritmo original de Breiman é feita escolhendo n vezes a partir de n pontos com substituição para calcular as estimativas das árvores individuais, ou seja, utilizando o método *bootstrap*. Embora uma das maiores vantagens do *bootstrap* é sua simplicidade, a teoria é complexa. De fato, a distribuição da amostra *bootstrap* $\mathcal{D}_n^\star$ é diferente daquela do conjunto original $\mathcal{D}_n$, como será mostrado a seguir. Assuma que $\mathbf{X}$ tem uma densidade, e note que em qualquer momento que os pontos de dados sejam amostrados com substituição então, com probabilidade positiva, ao menos uma observação do conjunto de dados original é selecionado mais de uma vez. Portanto, com probabilidade positiva, existem dois pontos de dados idênticos no conjunto $\mathcal{D}_n^\star$, de forma que a distribuição $\mathcal{D}_n^\star$ não pode ser absolutamente contínua.

O papel do *bootstrap* nas florestas aleatórias é ainda pouco compreendido e, atualmente, a maioria das análises acabam substituindo o *bootstrap* por algum método de subamostragem, assumindo que cada árvore é desenvolvida com $a_n < n$ exemplos aleatoriamente escolhidos sem substituição a partir do conjunto inicial. Na maioria das vezes, a taxa de subamostragem a_n/n tende a zero a alguma taxa estabelecida - uma consideração que exclui o método *bootstrap*. Nesse sentido, a análise das florestas aleatórias medianas feita pelo (Scornet, 2016) fornece alguma compreensão sobre a função e importância da subamostragem. No modelo de floresta aleatória mediana, uma vez que a direção de divisão é escolhido, o corte é feito na mediana empírica de $\mathbf{X}_i$ na célula. Em adição, a construção não para no nível k (onde k é a quantidade de vezes que cada célula é cortada), mas continua até cada célula conter exatamente uma observação. Desde que o número de casos que sobram nas folhas não cresce com n, cada árvore de uma floresta mediana é, em geral, inconsistente (inconsistência remete ao fato de que o erro do estimador não converge a zero a medida que o tamanho do conjunto de treino tende a infinito) (Biau; , 2016), (Györfi et al., 2002). Entretanto, (Scornet, 2016) mostra que se $a_n/n \to 0$, então a floresta mediana é consistente, mesmo que as árvores individuais não sejam. A consideração $a_n/n \to 0$ garante que todo par de observação $(\mathbf{X}_i, Y_i)$ é usado na construção da j-ésima árvore com uma probabilidade que se torna menor proporcionalmente ao crescimento de n. Isso força o ponto de consulta $\mathbf{x}$ a ser desconectado de $(\mathbf{X}_i, Y_i)$ em grande parte das árvores. Certamente, se esse não fosse o caso, então o valor predito no ponto $\mathbf{x}$ seria extremamente influenciado pelo par $(\mathbf{X}_i, Y_i)$, o que faria o modelo conjunto ser inconsistente. De fato, o erro de estimação da floresta mediana é pequeno se a probabilidade de conexão entre o ponto de consulta e todas as observações é pequena. Portanto, assumir que $a_n/n \to 0$ é uma forma aceitável para controlar essas probabilidades, garantindo que as partições são suficientemente diferentes (Biau; , 2016).

2.5.4 GBR - *Gradient Boosting Regressor*

2.5.4.1 Estimação de função

Um procedimento comum é restringir $F(\mathbf{x})$ para ser uma classe parametrizada de funções $F(\mathbf{x};\mathbf{P})$, onde $\mathbf{P} = \{P_1, P_2, ...\}$ é um conjunto finito de parâmetros cujos valores identificam membros individuais (diferentes) da classe. Nesta subseção, o foco será em expansões aditivas da forma

$$F(\mathbf{x};\{\beta_m, \mathbf{a}_m\}_1^M) = \sum_{m=1}^{M} \beta_m \cdot h(\mathbf{x};\mathbf{a}_m) \tag{56}$$

A função $h(\mathbf{x};\mathbf{a})$ na Equação (56) é normalmente uma simples função parametrizada das variáveis de entrada $\mathbf{x}$, caracterizada pelos parâmetros $\mathbf{a} = \{a_1, a_2, ...\}$. Os termos individuais no somatório acima diferem de acordo com os conjuntos de valores $\mathbf{a}_m$ escolhidos. Expansões como mostradas na Equação (56) estão no coração de tantos métodos de aproximação de funções como redes neurais, MARS, *wavelets* e *support vector machines*. O caso apresentado nessa subseção visualiza cada função $h(\mathbf{x},\mathbf{a}_m)$ como uma pequena árvore de regressão, tais como aquelas produziadas com o algoritmo CART. Para uma árvore de regressão, os parâmetros $\mathbf{a}_m$ são as variáveis de divisão e localização das divisões das árvores individuais, por exemplo (Friedman, 2001).

2.5.4.2 Otimização numérica

Em geral, escolher um modelo parametrizado $F(\mathbf{x};\mathbf{P})$ muda o problema de otimização de função para um problema de otimização de parâmetro

$$\mathbf{P}^* = \arg\min_{\mathbf{P}} \Phi(\mathbf{P}) \tag{57}$$

onde

$$\Phi(\mathbf{P}) = E_{y,\mathbf{x}} L(y, F(\mathbf{x};\mathbf{P})) \tag{58}$$

logo,

$$F^*(\mathbf{x}) = F(\mathbf{x};\mathbf{P}^*). \tag{59}$$

Lembrando que $\arg\min_{\mathbf{P}} \Phi(\mathbf{P})$ é interpretado como: qual valor ótimo $\mathbf{P}$ minimiza a função $\Phi(\mathbf{P})$ (Boyd; Vandenberghe, 2004). Para a maior parte de $F(\mathbf{x};\mathbf{P})$ e L, métodos de otimização numérica devem ser aplicados para resolver a Equação (57). Isso frequentemente envolve expressar a solução para os parâmetros na forma

$$\mathbf{P}^* = \sum_{m=0}^{M} \mathbf{p}_m \tag{60}$$

onde $\mathbf{p}_0$ é um valor inicial escolhido arbitrariamente e $\{\mathbf{p}_m\}_1^M$ são incrementos sucessivos ("passos"), onde cada um desses incrementos dependem da sequência de passos anteriores. A forma de calcular/obter cada valor $\mathbf{p}_m$ é definido pelo método de otimização escolhido (Friedman, 2001).

2.5.4.3 Gradiente descendente

Gradiente descendente é um dos mais simples métodos de minimização numérica utilizados. Esse método define $\{\mathbf{p}_m\}_1^M$ como segue. Primeiro, o gradiente atual $\mathbf{g}_m$ é calculado

$$\mathbf{g}_m = \{g_{jm}\} = \left\{ \left[\frac{\partial \Phi(\mathbf{P})}{\partial P_j} \right]_{\mathbf{P}=\mathbf{P}_{m-1}} \right\} \tag{61}$$

onde

$$\mathbf{P}_{m-1} = \sum_{i=1}^{m-1} \mathbf{p}_i \tag{62}$$

O passo é atualizado da seguinte forma

$$\mathbf{p}_m = -\rho_m \mathbf{g}_m \tag{63}$$

onde

$$\rho_m = \arg\min_{\rho} \mathbf{\Phi}\big(\mathbf{P}_{m-1} - \rho \mathbf{g}_m\big) \tag{64}$$

O gradiente negativo $-\mathbf{g}_m$ define a direção "gradiente descendente" e a Equação (64) é a "linha de busca" ao longo daquela direção.

2.5.4.4 Otimização numérica no espaço funcional

A análise inicia-se com uma aproximação "não paramétrica" e aplica-se a otimização numérica no espaço funcional. Isto é, considera-se $F(\mathbf{x})$ avaliada em cada ponto $\mathbf{x}$ sendo um "parâmetro" e busca-se minimizar

$$\mathbf{\Phi}(F) = E_{y,\mathbf{x}} L(y, F(\mathbf{x})) = E_{\mathbf{x}}[E_y(L(y, F(\mathbf{x}))|\mathbf{x}], \tag{65}$$

de forma que,

$$\phi(F(\mathbf{x})) = E_y[L(y, F(\mathbf{x}))|\mathbf{x}] \tag{66}$$

em cada $\mathbf{x}$ individual, com respeito a $F(\mathbf{x})$. No espaço funcional há um número infinito de tais parâmetros, mas em conjuntos de dados, apenas um número finito $\{F(\mathbf{x}_i)\}_1^N$ é envolvido. Seguindo o paradigma da otimização numérica, toma-se a solução como

$$F^*(\mathbf{x}) = \sum_{m=1}^{M} f_m(\mathbf{x}) \tag{67}$$

onde $f_0(\mathbf{x})$ é um valor inicial escolhido arbitrariamente, e $\{f_m(\mathbf{x})\}_1^M$ são funções incrementais definidas pelo método de otimização.

Para o gradiente descendente

$$f_m(\mathbf{x}) = -\rho_m g_m(\mathbf{x}) \tag{68}$$

com

$$g_m(\mathbf{x}) = \left[\frac{\partial \phi(F(\mathbf{x}))}{\partial F(\mathbf{x})}\right]_{F(\mathbf{x})=\mathbf{F}_{m-1}(\mathbf{x})} = \left[\frac{\partial E_y[L(y,F(\mathbf{x}))|\mathbf{x}]}{\partial F(\mathbf{x})}\right]_{F(\mathbf{x})=\mathbf{F}_{m-1}(\mathbf{x})} \tag{69}$$

e

$$F_{m-1}(\mathbf{x}) = \sum_{i=0}^{m-1} f_i(\mathbf{x}). \tag{70}$$

Assumindo regularidade suficiente que pode alternar diferenciação e integração, resulta na seguinte equação

$$g_m(\mathbf{x}) = E_y\left[\frac{\partial [L(y,F(\mathbf{x}))]}{\partial F(\mathbf{x})}|\mathbf{x}\right]_{F(\mathbf{x})=\mathbf{F}_{m-1}(\mathbf{x})} \tag{71}$$

O multiplicador ρ_m na Equação (68) é dado pelo seguinte método "line search" (Friedman, 2001)

$$\rho_m = \arg\min_{\rho} E_{y,\mathbf{x}} L(y, F_{m-1}(\mathbf{x}) - \rho g_m(\mathbf{x})). \tag{72}$$

onde métodos "line search" são do tipo onde há uma escolha de direção p_k e buscam ao longo dessa direção, a partir da iteração atual x_k para uma nova iteração com um valor mais baixo da função, limitando o número de tentativas de comprimentos do passo que será dado naquela direção, até encontrar um que se aproxima do mínimo, sem necessariamente buscar a minimização exata por ser cara e normalmente desnecessária (Wright; Nocedal et al., 1999).

2.5.4.5 Dados finitos

Essa aproximação não paramétrica se desfaz quando a distribuição conjunta de $(y,\mathbf{x})$ é estimada de um conjunto de dados finitos $\{y_i,\mathbf{x}_i\}_1^N$. Neste caso, $E_y[\cdot|\mathbf{x}]$ não pode ser estimada precisamente pelo valor de seu dado em cada $\mathbf{x}_i$, e mesmo se pudesse, poderia ser interessante estimar $F^*(\mathbf{x})$ em valores de $\mathbf{x}$ além dos pontos de dados de treino. A otimização de parâmetros pode ser feita através da seguinte equação que minimiza a estimativa da perda esperada baseada nos dados correspondentes (Friedman, 2001)

$$\{\beta_m,\mathbf{a}_m\}_1^M = \arg\min_{\{\beta'_m,\mathbf{a}'_m\}_1^M} \sum_{i=1}^N L\left(y_i, \sum_{m=1}^M \beta'_m h(\mathbf{x}_i;\mathbf{a}'_m)\right). \tag{73}$$

Em situações onde essa equação é infactível, pode-se tentar a seguinte aproximação

$$(\beta_m,\mathbf{a}_m) = \arg\min_{\beta,\mathbf{a}} \sum_{i=1}^N L(y_i, F_{m-1}(\mathbf{x}_i) + \beta h(\mathbf{x}_i,\mathbf{a})) \tag{74}$$

de forma que

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \beta_m(\mathbf{x};\mathbf{a}_m). \tag{75}$$

Em processamento de sinal, essa estratégia é chamada "busca correspondente" onde $L(y,F)$ é a perda de erro quadrático e a $\{h(\mathbf{x};\mathbf{a}_m)\}_1^M$ são funções "basis". Em aprendizagem de máquina, a Equação (74) é chamada "boosting" onde $y \in \{-1,1\}$ e $L(y,F)$ é uma perda exponencial e^{-yF} ou uma distribuição logarítmica binomial negativa. A função $h(\mathbf{x};\mathbf{a})$ é um modelo fraco ou modelo base, e normalmente é uma árvore de classificação ou regressão (Friedman, 2001).

Suponha que para uma perda particular $L(y,F)$ e/ou modelo base $h(\mathbf{x};\mathbf{a})$, a solução da Equação (74) é difícil de obter. Dado qualquer aproximador $F_{m-1}(\mathbf{x})$, a função $\beta_m h(\mathbf{x};\mathbf{a}_m)$ pode ser visualizada como o melhor passo em direção ao passo baseado na estimativa de $F^*(\mathbf{x})$, sob a restrição que a "direção" $h(\mathbf{x};\mathbf{a}_m)$ do passo seja um um membro da classe parametrizada de funções $h(\mathbf{x};\mathbf{a})$. Ela pode então ser considerada como um passo do gradiente descendente sob aquela restrição. O análogo baseado em dados do gradiente negativo sem restrição

$$-g_m(\mathbf{x}_i) = \left[\frac{\partial[L(y_i,F(\mathbf{x}_i))]}{\partial F(\mathbf{x}_i)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})} \tag{76}$$

fornece a melhor direção do passo do gradiente descendente $-\mathbf{g}_m = \{-g_m(\mathbf{x}_i)\}_1^N$ no espaço de dados N-dimensional em $F_{m-1}(\mathbf{x})$. Entretanto, esse gradiente é definido apenas em pontos de dados $\{\mathbf{x}_i\}_1^N$ e não pode ser generalizado para outros valores $\mathbf{x}$ da forma que está. Uma possibilidade para generalização é escolher o membro da classe parametrizada $h(\mathbf{x};\mathbf{a}_m)$ que produz $\mathbf{h}_m = \{h(\mathbf{x}_i;\mathbf{a}_m)\}_1^N$ mais paralelo a $-\mathbf{g}_m \in \mathbb{R}^N$. Este é o $h(\mathbf{x};\mathbf{a})$ com mais alta correlação com $-g_m(\mathbf{x})$ sobre a distribuição de dados, podendo ser obtido a partir da solução

$$\mathbf{a}_m = \arg\min_{\mathbf{a},\beta} \sum_{i=1}^{N} [-g_m(\mathbf{x}_i) - \beta h(\mathbf{x}_i;\mathbf{a})]^2 \tag{77}$$

Este gradiente negativo com restrição $h(\mathbf{x};\mathbf{a}_m)$ é utilizado no lugar do sem restrição $-g_m(\mathbf{x})$ no método do gradiente descendente. Mais especificamente, a busca é feita pela seguinte equação

$$\rho_m = \arg\min_{\rho} \sum_{i=1}^{N} L(y_i, F_{m-1}(\mathbf{x}_i) + \rho h(\mathbf{x}_i;\mathbf{a}_m)) \tag{78}$$

e a aproximação atualizada por

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m h(\mathbf{x};\mathbf{a}_m). \tag{79}$$

Basicamente, em vez de obter a solução sob uma restrição suavizada (Equação (74)), a restrição é aplicada para a solução sem restrição através do ajuste de $h(\mathbf{x};\mathbf{a})$ para as pseudo-respostas $\{\tilde{y}_i = -g_m(\mathbf{x}_i)\}_{i=1}^N$ (Equação (71)). Isso permite a substituição do difícil problema de minimização da função (74) usando a minimização da função "mínimo quadrado (*least-squares*)" (Equação (77)), seguida por apenas um único parâmetro de otimização baseado no critério original (Equação

(78)). Portanto, para qualquer $h(\mathbf{x};\mathbf{a})$ para o qual existe um algoritmo "mínimo quadrado" factível para resolver a Equação (77), pode-se usar essa aproximação para minimizar qualquer perda diferenciável $L(y,F)$ em conjunto com modelagem aditiva. Essa abordagem leva ao seguinte algoritmo usando gradiente descendente. É importante destacar que qualquer critério de ajuste

Algoritmo 3: Gradient_Boost

$F_0(\mathbf{x}) = \arg\min_\rho \sum_{i=1}^{N} L(y_i,\rho)$
for $m = 1$ *to* M **do**
  $\tilde{y}_i = -\left[\frac{\partial L(y_i,F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}, \quad i = 1,\ldots,N$
  $\mathbf{a}_m = \arg\min_{\mathbf{a},\beta} \sum_{i=1}^{N} [\tilde{y}_i - \beta h(\mathbf{x}_i;\mathbf{a})]^2$
  $\rho_m = \arg\min_\rho \sum_{i=1}^{N} L(y_i, F_{m-1}(\mathbf{x}_i) + \rho h(\mathbf{x}_i;\mathbf{a}_m))$
  $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m h(\mathbf{x};\mathbf{a}_m)$
finaliza_**for**
Finaliza o Algoritmo

que estima o valor esperado condicional (dado $\mathbf{x}$) poderia, a princípio, ser utilizado para estimar o gradiente negativo (Equação (71)) na linha 4 do Algoritmo 3. *Least-squares* (Equação (77)) foi escolhido pelas suas propriedades computacionais superiores quando comparado com outros algoritmos *least-squares* (Friedman, 2001).

2.5.4.6 Árvores de regressão (GBR)

Considera-se o caso onde cada modelo base é um nó terminal J de uma árvore de regressão. Cada modelo de árvore de regressão tem a forma aditiva

$$h(\mathbf{x};b_j,R_1^J) = \sum_{j=1}^{J} b_j 1(\mathbf{x} \in R_j). \tag{80}$$

$\{R_j\}_1^J$ são regiões disjuntas que coletivamente cobrem o espaço de todos os valores conjuntos das variáveis preditoras $\mathbf{x}$. Essas regiões são representadas pelos nós terminais da árvore correspondente. A função indicadora assume valor um se seu argumento é verdadeiro e zero para os outros casos. Os "parâmetros" desse modelo base são os coeficientes $\{b_j\}_1^J$ e as quantidades que definem as fronteiras das regiões $\{R_j\}_1^J$. Essas são as variáveis de divisão, e os valores daquelas variáveis que representam as divisões em nós não terminais da árvore. Por conta das regiões serem disjuntas, a Equação (80) é equivalente à regra de predição: se $\mathbf{x} \in R_j$, então $h(\mathbf{x}) = b_j$ (Friedman, 2001). Para uma árvore de regressão, a atualização na linha 6 do Algoritmo 3 se torna

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m \sum_{j=1}^{J} b_{jm} 1(\mathbf{x} \in R_{jm}) \tag{81}$$

$\{R_{jm}\}_1^J$ são as regiões definidas pelos j-ésimos nós terminais da árvore na m-ésima iteração. Elas são construídas para prever as pseudo-respostas $\{\tilde{y}_i\}_1^N$ (linha 3) pelos mínimos quadrados (linha

4). Os coeficientes $\{b_{jm}\}$ são os coeficientes mínimos quadrados correspondentes

$$b_{jm} = \text{média}_{\mathbf{x}_i \in R_{jm}} \tilde{y}_i. \tag{82}$$

A atualização (Equação (81)) pode ser alternativamente expressa como

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \sum_{j=1}^{J} \gamma_{jm} 1(\mathbf{x} \in R_{jm}) \tag{83}$$

com $\gamma_{jm} = \rho_m b_{jm}$. Devido à natureza distinta das regiões produzidas pelas árvores de regressão, os coeficientes γ_{jm} são dados por

$$\gamma_{jm} = \arg\min_{\gamma} \sum_{\mathbf{x}_i \in R_{jm}} L(y_i, F_{m-1}(\mathbf{x}_i) + \gamma). \tag{84}$$

De forma que esta é a atualização ótima em cada região contendo $\mathbf{x}_i$ do nó terminal, baseado na função perda L, dada a aproximação atual $F_{m-1}(\mathbf{x})$. Para o caso do desvio absoluto mínimo, a Equação (84) se torna

$$\gamma_{jm} = \text{mediana}_{\mathbf{x}_i \in R_{jm}} \{y_i - F_{m-1}(\mathbf{x}_i)\}, \tag{85}$$

que é simplesmente a mediana dos resíduos atuais no j-ésimo nó na m-ésima iteração. Em cada iteração, uma árvore de regressão é construída para prever a função *sign* dos resíduos atuais $y_i - F_{m-1}(\mathbf{x}_i)$, baseado em um critério de mínimos quadrados. A aproximação é atualizada pela adição dos resíduos em cada um dos nós terminais derivados. As pseudo-respostas $\tilde{y}_i$ tem apenas dois valores, $\tilde{y}_i \in \{-1, 1\}$. O algoritmo é então escrito como apresentado em 4 (Friedman, 2001).

Algoritmo 4: LAD_TreeBoost

$F_0(\mathbf{x}) = \text{mediana}\{y_i\}_1^N$;
for $m = 1$ *to* M **do**
  $\tilde{y}_i = \text{sign}(y_i - F_{m-1}(\mathbf{x}_i))$, $i = 1, N$;
  $\{R_{jm}\}_1^J = J$-terminal node tree$(\{\tilde{y}_i, \mathbf{x}_i\}_1^N)$;
  $\gamma_{jm} = \text{median}_{\mathbf{x}_i \in R_{jm}} \{y_i - F_{m-1}(\mathbf{x}_i)\}$, $j = 1, J$;
  $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \sum_{j=1}^{J} \gamma_{jm} \mathbb{1}(\mathbf{x} \in R_{jm})$;
finaliza_**for**
Finaliza o Algoritmo

2.5.5 XGBoost (*eXtreme Gradient Boosting*)

XGBoost é um sistema de aprendizado de máquina escalável para reforço (*boost*) de algoritmos "árvores". Esse algoritmo é disponível em código aberto, garantindo várias vitórias em competições no site *Kaggle* e na *KDDCup 2015*, por exemplo. A *KDDCup 2015* foi onde o

poder do *XGBoost* tornou-se notório, onde cada time vencedor no *top-10* utilizou este algoritmo. Além do que, os times vencedores comentaram que modelos em conjunto mostraram pouca otimização quando comparados ao *XGBoost* sozinho. O fator mais importante por trás do sucesso do XGBoost é sua escalabilidade em aplicações de diversos segmentos diferentes, de forma que o algoritmo roda mais de 10 vezes mais rápido que outros algoritmos populares existentes em uma única máquina (Chen; Guestrin, 2016).

2.5.5.1 Objetivo de aprendizado regularizado

Para um dado conjunto de dados com n exemplos e m características $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}(|\mathcal{D}| = n, \mathbf{x}_i \in \mathbb{R}^m, y_i \in \mathbb{R})$, um modelo de árvores em conjunto (mostrado na Figura 4) usa K funções aditivas para prever a saída.

$$\hat{y}_i = \phi(\mathbf{x}_i) = \sum_{k=1}^{K} f_k(\mathbf{x}_i), f_k \in \mathcal{F}, \tag{86}$$

onde $\mathcal{F} = \{f(\mathbf{x}) = w_{q(\mathbf{x})}(q : \mathbb{R}^m \to T, w \in \mathbb{R}^T)\}$ é o espaço das árvores de regressão. q representa a estrutura de cada árvore que mapeia um exemplo para o índice da folha correspondente. T é o número de folhas em cada árvore. Cada f_k corresponde a uma árvore independente com estrutura q e pesos w para cada folha. Diferente de árvores de decisão, cada árvore de regressão contém uma pontuação contínua em cada folha, usa-se w_i para representar a pontuação na i-ésima folha.

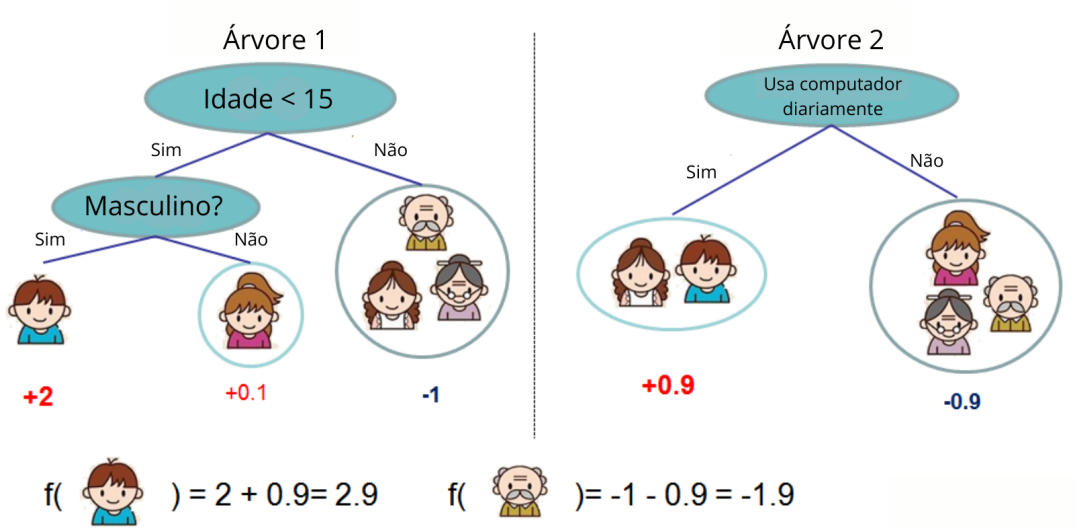

Figura 4 – Modelo de árvores em conjunto. A previsão final para um dado exemplo é a soma das previsões de cada árvore. Adaptado de (Chen; Guestrin, 2016).

Para aprender o conjunto de funções usadas no modelo, minimiza-se o seguinte objetivo regularizado:

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k) \tag{87}$$

onde $\Omega(f) = \gamma \cdot T + 0,5 \cdot \lambda ||w||^2$. A variável l representa uma função perda convexa diferenciável que calcula a diferença entre a previsão $\hat{y}_i$ e o valor alvo y_i. O segundo termo Ω penaliza a complexidade do modelo.

Além do objetivo regularizado, duas técnicas adicionais são utilizadas para prevenir *overfitting*. Sendo a primeira técnica a contração introduzida por Friedman, essa técnica escala pesos recém criados por um fator η após cada passo de *boosting* da árvore. Similar a uma taxa de aprendizagem em otimização estocástica, contração reduz a influência de cada árvore individual e deixa espaço para árvores futuras melhorarem o modelo. A segunda técnica é subamostragem de colunas (características). Essa técnica é também utilizada no algoritmo RF, e além de prevenir *overfitting*, também acelera paralelismo do algoritmo (Chen; Guestrin, 2016).

2.5.6 CatBoost

CatBoost é uma biblioteca de código aberto que lida de forma satisfatória com características categóricas, tendo uma implementação GPU de algoritmo de aprendizagem e uma implementação CPU de algoritmo de pontuação, superando algoritmos como *XGBoost* e *LightGBM* em algoritmos em conjunto de tamanhos similares (Dorogush; Ershov; Gulin, 2018).

CatBoost, como outras implementações padrões de gradiente *boosting*, constrói cada nova árvore para aproximar os gradientes do modelo atual. Entretanto, todos os algoritmos clássicos *boosting* sofrem por *overfitting* causado pelo problema de estimações do gradiente ponto a ponto com viés, de forma que *CatBoost* utiliza uma modificação do algoritmo gradiente *boosting* para tentar resolver esse problema (Dorogush; Ershov; Gulin, 2018).

Sendo F^i o modelo construído após construir as primeiras i árvores, $g^i(\mathbf{X}_k, Y_k)$ sendo o valor do gradiente na k-ésima amostra de treino após construir i árvores. Para fazer o gradiente $g^i_{(\mathbf{X}_k, Y_k)}$ não ter tendência com respeito ao modelo F^i, é necessário ter F^i treinado sem a observação $\mathbf{X}_k$. Desde que são necessários gradientes sem viéses para todas as amostras de treino, nenhuma amostra poderia ser utilizada para treinar F^i, o que aparentemente faria o processo de treino infactível. Considera-se então o seguinte truque para lidar com este problema: para cada exemplo $\mathbf{X}_k$, treina-se um modelo M_k separadamente que nunca é atualizado utilizando a estimativa do gradiente para esse exemplo. Com M_k, estima-se o gradiente em $\mathbf{X}_k$ e usa-se essa estimativa para pontuar a árvore resultante. No algoritmo *CatBoost* são geradas s permutações aleatórias do conjunto de treino. Usa-se muitas destas permutações para melhorar a robustez do algoritmo: amostra-se uma permutação aleatória e obtêm-se os gradientes nesta. Usa-se diferentes permutações para treinar modelos diferentes, de forma que não acontece *overfitting* por conta das permutações. Além disso, *CatBoost* usa árvores *oblivious* como base do modelo, tais árvores usam o mesmo critério de divisão em níveis inteiros da árvore, sendo balanceadas e menos propensas a *overfitting* (Dorogush; Ershov; Gulin, 2018).

2.5.7 Otimização de hiperparâmetros

Hiperparâmetros são parâmetros que devem ser definidos antes de executar um algoritmo de aprendizagem de máquina, opostos aos parâmetros normais de um algoritmo, que não são fixados antes da execução, e sim otimizados enquanto o algoritmo é treinado. Alguns exemplos de hiperparâmetros são o número de variáveis que são consideradas em cada divisão em uma floresta aleatória, o número de passos *boost* no gradiente *boosting*, ou o número k no algoritmo k-vizinhos mais próximos (Probst, 2019).

O processo de encontrar hiperparâmetros ótimos para um algoritmo de aprendizagem para um determinado conjunto de dados é chamado de *tuning* (afinação/sintonização/ajuste ou simplesmente otimização de hiperparâmetro), e o valor ótimo pode se referir a diferentes medidas de performance (por exemplo, a taxa do erro ou a área sob a curva (AUC - area under curve)) ou ao tempo de execução (no menor tempo possível ou ao menos em um período razoável) (Probst; Wright; Boulesteix, 2019), (Probst, 2019). Infelizmente, a relação entre a performance dos algoritmos de aprendizagem de máquina e hiperparâmetros não é nítida, então, na prática, é necessário continuamente ajustar os hiperparâmetros e treinar um número de modelos com diferentes combinações de valores, e então comparar a performance dos modelos para melhor escolher o melhor modelo (Wu et al., 2019). As duas estratégias mais simples são busca em grade (*grid search*), na qual todas as combinações possíveis do espaço de parâmetros dado são avaliadas, e busca aleatória (*random search*), na qual valores de hiperparâmetros são utilizados aleatoriamente a partir de um espaço de hiperparâmetros especificado (Probst, 2019). Por conta dessas técnicas citadas serem simples, o custo computacional é extremamente alto, além do mais, deve-se ter bom conhecimento e/ou experiência para selecionar o espaço de busca. Processos de afinamento (*tuning*) são cada vez mais executados através de métodos automatizados que tentam encontrar os hiperparâmetros ótimos em menos tempo usando um busca informada baseada em alguma estratégia, tal que operações manuais adicionais são desnecessárias. Entre os métodos mais recentes, Otimização Bayesiana é uma das melhores escolhas para otimizar a função objetivo (Zhang et al., 2021).

Otimização Bayesiana (BO - *Bayesian Optimization*) é uma estratégia eficiente para otimizar uma função desconhecida ou *black-box* (caixa-preta), mostrando seu potencial em funções que são computacionalmente caras (Wu et al., 2019), (Chaibi et al., 2021). A Otimização Bayesiana consiste de dois componentes principais: um modelo probabilístico $p(f)$ para a função objetivo $f(\mathbf{x})$, e uma função aquisição a para explorar o modelo e decidir onde será a próxima amostragem (qual o próximo valor de um dado hiperparâmetro que será utilizado a fim de obter melhores resultados) (Frazier, 2018), (Snoek; Larochelle; Adams, 2012). Regressão por Processos Gaussianos (GP - *Gaussian Processes*) são uma escolha proeminente para $p(f)$, sendo o modelo probabilístico mais utilizado (Klein et al., 2017). A filosofia na BO é utilizar toda a informação disponível das avaliações/iterações anteriores de $f(\mathbf{x})$ em vez de simplesmente confiar em gradientes locais ou aproximações (Snoek; Larochelle; Adams, 2012). Isso resulta

em um método robusto que consegue encontrar o mínimo de uma função caixa-preta com relativamente poucas avaliações/iterações, de forma que o alto custo está normalmente em determinar o próximo ponto a ser avaliado, exceto quando avaliações de $f(\mathbf{x})$ são "caras" para serem realizadas ao treinar algoritmos de aprendizado (Snoek; Larochelle; Adams, 2012), (Klein et al., 2017).

2.5.8 Otimização de parâmetros

Otimização por enxame de partículas é um método baseado em população, autoadaptativo e estocástico que determina a melhor posição e avalia o valor ótimo de uma função para cada partícula em cada posição. Baseado nas velocidades atuais, cada posição ótima da partícula e cada posição ótima dos vizinhos, este algoritmo determina novas velocidades. Para manter as partículas dentro de seus limites, as localizações, velocidades e vizinhos mudam periodicamente. O algoritmo continua até ele chegar em um ponto de parada determinado pelo tempo ou quantidade de iterações, por exemplo. Na otimização por enxame de partículas, cada solução é descrita como um "pássaro/partícula" em um espaço de busca. Estas partículas movem-se em um enxame para a posição ótima. A partícula carrega tanto o vetor de posição como o de velocidade em um problema dado por um espaço n-dimensional. A posição da partícula j é denotada por $X_j = (X_{j1}, X_{j2}, ..., X_{jn})$ e a velocidade da partícula j é representada como $V_j = (V_{j1}, V_{j2}, ..., V_{jn})$. A velocidade e a posição de cada partícula é atualizada pelas Equações (88) e (89).

$$V_{ij}^{k+1} = V_{ij}^{k} + C_p \Gamma_p (X_{pbestij}^{k} - X_{ij}^{k}) + C_g \Gamma_g (X_{gbestij}^{k} - X_{ij}^{k}) \tag{88}$$

$$X_{ij}^{k+1} = X_{ij}^{k} + V_{ij}^{k+1}, \text{i = 1,2, ..., N e j = 1, 2, ..., P} \tag{89}$$

onde $X_{pbestij}$ = melhor posição individual da partícula; X_{gbest} = melhor posição global do grupo; C_g, C_p = coeficiente de aceleração social e cognitivo; N = número total de variáveis; Γ = número aleatório distribuído uniformemente; V_{ij}^{k+1} = velocidade da j-ésima partícula da i-ésima variável na (k+1)-ésima iteração (Jain; Nangia; Jain, 2018).

2.6 CONJUNTOS DE DADOS

2.6.1 Tamanho do conjunto de dados (*Dataset*)

Em aprendizagem de máquina, a performance de generalização é extremamente importante, isto é, a performance dos algoritmos utilizados em realizar previsões em dados novos e não vistos durante a fase de treino do algoritmo (Viering; Loog, 2022), e para isso, o tamanho da amostra é um tópico que deve ser considerado. Em Vabalas et al. (2019), simulações mostraram que os modelos de previsão testados foram capazes de aprender um padrão nos dados mais eficientemente e conquistar uma maior performance com conjuntos de dados maiores. Em Ng et

al. (2020), modelos mais simples tiveram resultados melhores em conjuntos de dados menores (com menos de 1400 dados), enquanto modelos mais robustos tiveram resultados melhores quando o conjunto de dados era maior (mais que 2500 dados). Uma informação importante citada por vários autores é que existe um limite da melhoria de performance de um modelo com o aumento do tamanho da amostra, por exemplo, o aumento na precisão de modelos de aprendizagem de máquina se torna insignificante quando o número de amostras é maior que 5000 (Ng et al., 2020). Há também algoritmos que estimam o tamanho da amostra necessária como apresentado em Figueroa et al. (2012), onde o algoritmo utiliza curvas de aprendizado e teve resultados melhores do que algoritmos descritos em literaturas anteriores em relação a medidas de ajuste. Em Althnian et al. (2021), os resultados revelaram que tanto para grandes conjuntos de dados quanto para pequenos, construir um conjunto de dados que seja bem representativo em relação à distribuição original, em vez de focar no tamanho, tem um maior impacto na performance geral dos modelos de classificação.

2.6.2 Seleção de características

Toda etapa é importante ao usar algoritmos de ML, e isso inclui a seleção de características e variáveis em criar o conjunto de dados. Alguns benefícios de seleção de características e variáveis são: facilitar a visualização de dados e compreensão dos dados, reduzir tempo de treino, requerimentos de armazenamento e medição, e contornar a "maldição da dimensionalidade" para melhorar a performance da predição. Selecionar as variáveis mais relevantes é normalmente subótimo para construir um modelo de previsão, principalmente se as variáveis são redundantes. Infelizmente, algumas vezes, essas variáveis redundantes são importantes para o modelo utilizado (Guyon; Elisseeff, 2003). As variáveis selecionadas para a predição da capacitância de supercapacitores na literatura, por exemplo, incluem: razão do ativador, temperatura de ativação, grau de grafitização, proporção de celulose, volume total dos poros, área superficial específica, densidade de corrente, agente de ativação utilizado, janela de potencial, volume de microporos, etc (Liao et al., 2024), (Rahimi; Abbaspour-Fard; Rohani, 2022), (Mishra et al., 2023).

2.6.3 Processamento dos dados

Após selecionar os dados que serão utilizados, esses dados devem passar através de um pré-processamento, que engloba tarefas como limpeza de dados, integração dos dados e transformação dos dados para serem utilizados pelos algoritmos de aprendizagem de máquina. Normalmente, os problemas mais complicados são os "*outliers*" (valores anormalmente altos ou baixos que estão fora da distribuição das variáveis (Kwak; Kim, 2017)), informação faltante ou perdida (Ishaq et al., 2024), que causam diferentes problemas como degradação de performance, problemas de análise de dados e resultados enviesados (Emmanuel et al., 2021). Para *outliers*, o método utilizado mais comum é removê-los como uma forma de aparar o conjunto de dados (Kwak; Kim, 2017). Para dados faltantes, o tratamento é normalmente mais complicado e envolve

o tipo de dado faltante (faltantes aleatórios, faltantes completamente aleatórios, faltantes não aleatórios (Cummings, 2013)) e qual técnica é escolhida para ser utilizada, podendo-se citar algumas opções: *missForest* utilizada em Alsaber, Pan e Al-Hurban (2021) teve o menor erro de imputação tanto para as variáveis categóricas quanto para as contínuas em várias taxas diferentes de ausência (variando de 5% a 40% no estudo realizado), eliminação é outra técnica que pode ser utilizada (mas esta pode introduzir tendência na análise), imputação *hot-deck* e métodos de imputação inspirados em algoritmos de aprendizagem de máquina (Emmanuel et al., 2021). O método SICE proposto em Khan e Hoque (2020) diminuiu o erro em torno de 11% através da imputação de dados numéricos e binários. Outro método para dados faltantes é a imputação de valor único, que ao invés de estimar um valor desconhecido de uma certa característica faz-se a substituição de cada valor faltante com um valor único arbitrário (Saar-Tsechansky; Provost, 2007), (Lee; Kim, 2010). Algumas vezes é necessário converter dados categóricos para dados numéricos dependendo do algoritmo a ser utilizado, para isso há alguns métodos como codificação *one hot*, codificação ordinal, codificação polinomial, etc (Potdar; Pardawala; Pai, 2017).

2.6.4 Métricas de modelos de classificação e regressão

Algumas métricas comumente utilizadas para avaliar algoritmos de classificação são: precisão, coeficiente kappa de Cohen (κ), raiz do erro quadrático médio (RMSE - *Root Mean Squared Error*), erro absoluto médio (MAE - *Mean Absolute Error*), taxa de verdadeiros positivos (TPR - *True Positive Rate*) e taxa de falsos positivos (FPR - *False Positive Rate*) (Chicco; Jurman, 2023).

$$\text{Precisão} = \frac{TP+TN}{TP+TN+FP+FN} \tag{90}$$

$$\kappa = \frac{p_o - p_e}{1 - p_e} \tag{91}$$

$$MAE = \frac{1}{K}\sum_{i=1}^{K} |T_i - P_i| \tag{92}$$

$$RMSE = \sqrt{\sum_{i=1}^{K} (T_i - P_i)^2 / K} \tag{93}$$

$$TPR = TP/TTP \tag{94}$$

$$FPR = FP/TTN \tag{95}$$

onde TP são os elementos verdadeiros positivos, TN são os elementos verdadeiros negativos, FN são os elementos falsos negativos (Grandini; Bagli; Visani, 2020); p_0 é a proporção de casos corretamente classificados, p_e é a proporção esperada de casos classificados corretamente por acaso (Foody, 2020); K denota o número de exemplos, P_i e T_i são os valores previsto e alvo para o i-ésimo exemplo, respectivamente (Qi et al., 2020), (Rocha; Cortez; Neves, 2007); TTP e TTN são os elementos totais positivos e negativos, respectivamente (Wang; Carvalho, 2024).

3 METODOLOGIA

Aqui serão explicadas duas abordagens para duas aplicações diferentes: uma das aplicações é a previsão e otimização da produção de gás de síntese em um processo utilizando biomassa, enquanto a outra aplicação foca na previsão da capacitância de supercapacitores.

3.1 METODOLOGIA PARA A PREVISÃO E OTIMIZAÇÃO DE GÁS DE SÍNTESE

O conjunto de dados pode ser construído através de experimentação ou utilizando resultados experimentais de outros estudos utilizando, por exemplo, o *software Plot Digitizer 2.6.8* ou *GetData Graph Digitizer 2.24*. Os conjuntos de dados aqui utilizados podem ser obtidos através do material suplementar de (Li et al., 2023), sendo 150 pontos de dados no conjunto de dados dos três produtos finais (onde os tipos de biomassa são variados) e 342 pontos de dados no conjunto da composição do gás de síntese. Caracaterísticas importantes da biomassa são teor de umidade (M), atributos de análise aproximada (*i.e.,* teores de carbono fixado (FC), de matéria volátil (VM) e de cinzas (*Ash*)), atributos de análise elementar (*i.e.*, teores de carbono (C), de hidrogênio (H), de oxigênio (O) e de nitrogênio (N)) e atributos de análise de teor relativo (*i.e.*, teores de celulose (Ce), de hemicelulose (He) e de lignina (Li)). Foram considerados dois tipos de saídas no desenvolvimento dos modelos de aprendizagem de máquina. Um dos conjuntos de saída é composto pelos rendimentos dos três subprodutos finais, biocarvão, alcatrão ou bio-óleo, e gás de síntese, enquanto que o outro conjunto é composto pelos rendimentos da composição do gás de síntese. Para melhorar a performance dos algoritmos de aprendizado de máquina é necessário um pré-tratamento do conjunto de dados para unificar unidades, retirar *outliers* (valores diferindo anormalmente dos outros), preenchimento de valores faltantes e divisão do conjunto. A normalização dos dados é sempre realizada como pré-tratamento essencial por conta do conjunto de dados sempre conter variáveis de diferentes faixas, médias e desvio padrão, sendo a variável normal padrão dada por:

$$Z_i = (X_i - \rho)/\sigma \tag{96}$$

onde X_i denota o dado da variável de entrada original, ρ é a média dos valores da variável e σ é o desvio padrão da variável (Dong et al., 2023).

Uma vez que os conjuntos de dados são preparados, cada um deles são aleatoriamente divididos em duas partes com a razão de 80:20, como uma estratégia comumente adotada (Li et al., 2023).

O coeficiente de correlação de Spearman (SCC - *Spearman Correlation Coefficient*) é utilizado para avaliar a correlação linear entre quaisquer duas variáveis. Utilizar o SCC no desenvolvimento de um modelo de aprendizagem de máquina é vantajoso por vários motivos: ele descreve relações não lineares entre variáveis, determina quais características são mais fortemente associadas com a saída e identifica valores atípicos (*outliers*), ou seja, variáveis que mostram fraca correlação com a variável alvo. O método para calcular o SCC é apresentado na

Equação (97) (Divine et al., 2024):

$$SCC = 1 - \frac{6 \cdot \sum d_i^2}{n(n^2-1)} \tag{97}$$

Os algoritmos utilizados nos trabalhos analisados deste tema incluem: *Random Forest*, *AdaBoost*, *XGBoost*, *Gradient Boosting Decision Tree* (GBDT), Perceptron multicamadas com algoritmo Levenberg-Marquardt, *Adaptive Neuro-Fuzzy Inference System* (ANFIS), *Radial Basis Function* (RBF), *Support Vector Machine* (SVM), *Gaussian Process Regression* (GPR), *Multi-Layer* Perceptron (MLP) e *Multi Linear Regression* (MLR) (Divine et al., 2024), (Ma et al., 2023), (Rahimi et al., 2023). Foram escolhidos os algoritmos RF e GBDT nesta aplicação por apresentarem melhores resultados em boa parte da literatura de processos termoquímicos com biomassa estudada. O método de validação escolhido foi o de validação cruzada *k-fold*. Por conta da escolha dos hiperparâmetros afetar enormemente a modelagem, para os algoritmos RF e GBDT, o número e profundidade das árvores de ambos foram otimizados, e mais dois hiperparâmetros (taxa de aprendizagem e taxa de subamostragem) foram também otimizados para o GBDT por conta de ser um algoritmo mais complicado que o RF (Li et al., 2023).

O algoritmo GBDT mostrou resultados superiores aos encontrados utilizando o RF. Logo, o GBDT foi utilizado para as predições dos produtos finais e a composição do gás de síntese. A importância característica é obtida através da redução média de impurezas. A impureza é quantificada pelo critério de divisão durante a construção das árvores de decisão. Para árvores de decisão individuais, a importância de uma característica poderia ser determinada pelo *rank* relativo da característica que mais contribui para os pontos de divisão de uma árvore. Portanto, se uma característica é mais frequentemente utilizada nos pontos de divisão, ela tem uma significância maior. Além do mais, esta noção de importância pode ser estendida para métodos conjuntos baseados em árvore (como o GBDT) através da média da importância característica de cada árvore. Adicionalmente, as correlações entre variáveis de entrada e variáveis alvo podem ser visualizados através de gráficos de dependência parcial (*Partial Dependence Plots* - PDP). Em termos de PDP baseados em árvore, se um nó de divisão envolve um dado ponto de uma característica de entrada, o ramo direito ou esquerdo correspondente ou ambos os ramos serão ponderados por uma fração das amostras de treino. A dependência parcial final é calculada por uma média ponderada dos valores de todas as folhas verificadas (Li et al., 2023). O algoritmo de otimização por enxame de partículas foi utilizado para a otimização de parâmetros do gás de síntese.

3.2 PREVISÃO DE CAPACITÂNCIA DE SUPERCAPACITORES

O conjunto de dados disponível em Ghosh, Rao e Thomas (2021) foi selecionado (esse conjunto de dados é formado pelas seguintes variáveis: qual o elemento principal do material, se o material é um óxido ou nitreto, se é um compósito ou não, a área superficial do material, qual a faixa do potencial elétrico, a densidade de corrente, substrato utilizado, e capacitância

específica. Mais detalhes sobre o conjunto de dados podem ser verificados na fonte original), e algumas alterações realizadas serão descritas aqui. Por conveniência, *Python* foi a linguagem de programação utilizada neste trabalho. Após selecionar o conjunto de dados, a biblioteca *Pandas* foi utilizada: nas colunas categóricas de variáveis independentes transformando estas em colunas codificadas do tipo *one-hot* (0 se não pertence a aquela categoria e 1 se pertence), e para plotar a correlação de *Spearman* através de mapa de calor (para o mapa de calor, a biblioteca *Pandas* foi utilizada em conjunto com as bibliotecas *Seaborn*, *NumPy* e *Matplotlib*. As funções principais utilizadas foram *heatmap* da biblioteca *Seaborn* e *corr* da Pandas) mostrado na Figura 5. A correlação de *Pearson* também foi analisada, mas sem resultados interessantes, de forma que não será mostrada. Correlação é uma métrica de associação entre pares de variáveis, onde essa métrica varia de -1 a +1, de forma que valores positivos indicam que duas variáveis tendem a aumentar ou diminuir "simultaneamente", enquanto que valores negativos indicam que se uma variável aumenta, a outra diminui e vice-versa (Schober; Boer; Schwarte, 2018), (Chok, 2010).

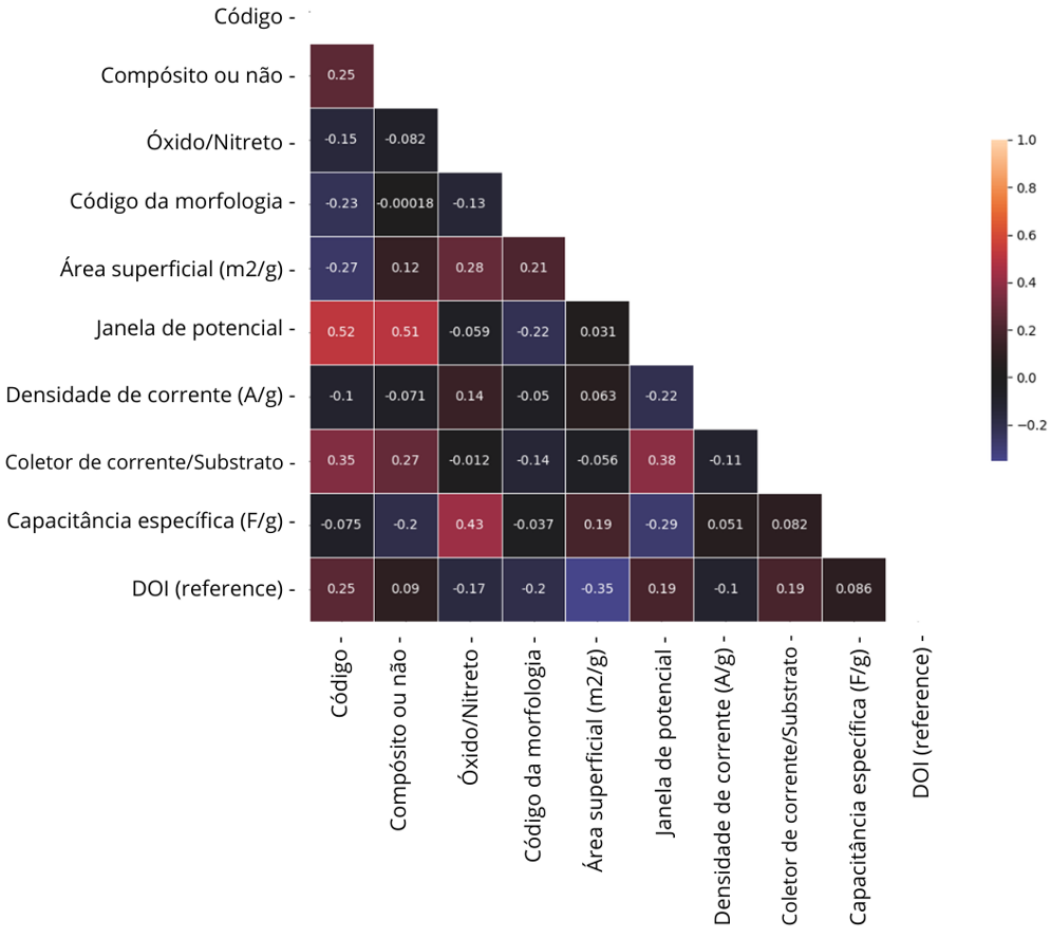

Figura 5 – Correlação de Spearman (autoria própria).

Após isso, a variável dependente (onde a variável dependente são valores de capacitância específica) foi transformada para categórica para o conjunto de dados ser utilizado com um algoritmo de classificação. Para se fazer uma previsão conjunta, usando o algoritmo de classificação para classificar cada linha (cada amostra de variáveis independentes) em diferentes grupos

de capacitância específica, enquanto um algoritmo de regressão também será utilizado (sem transformar a variável dependente em categórica) para prever o valor da capacitância específica exata de um ou mais casos escolhidos. Em Chen et al. (2019) e Pintea et al. (2023) há mais detalhes (em outros tipos de aplicações) sobre o método de utilizar classificação e regressão conjuntamente.

O algoritmo utilizado neste trabalho para classificação foi escrito usando *bootstrap*, validação cruzada estratificada e otimização Bayesiana como segue:

- (I) Define-se a faixa de valores para os hiperparâmetros para cada modelo a ser utilizada usando otimização Bayesiana (com a função *BayesianOptimization* através da biblioteca *Bayesian Optimization* (Nogueira, 2014–)). Essa faixa de valores é normalmente escolhida usando valores padrões definidos em artigos que utilizam o mesmo algoritmo de classificação (de preferência no mesmo tipo de aplicação ou em aplicações com tema muito próximo da sua pesquisa), a faixa de valores dos hiperparâmetros utilizada em cada algoritmo poderá ser visualizada no código disposto no Apêndice;
- (II) Para cada conjunto de hiperparâmetros, o algoritmo irá funcionar da seguinte forma:
- (II.A) Listas são criadas para armazenar os valores de cada métrica;
- (II.B) 10 conjuntos (*folds*) com o mesmo tamanho do conjunto de dados original são criados usando *bootstrap* (com a função *resample* da biblioteca *Scikit-learn*);
- (II.C) Usar validação cruzada estratificada (com a função *Stratified-KFold* da biblioteca *Scikit-learn* dentro de cada conjunto criado com *bootstrap* (de forma que cada conjunto tem (quase) o mesmo percentual de amostras de classes minoritárias e majoritárias como o conjunto de dados original (Szeghalmy; Fazekas, 2023)));
- (II.D) Divide cada conjunto *bootstrap* criado em 4 conjuntos de treino e 1 de teste;
- (II.E) Treina e avalia o modelo selecionado em cada *fold* (o *RandomForestClassifier* da biblioteca *Scikit-learn* e o *XGBClassifier* da biblioteca *XGBoost* foram utilizados);
- (II.F) As métricas são então calculadas para cada *fold* e uma média é calculada;
- (III) O conjunto de hiperparâmetros com a melhor precisão é então escolhido;
- (IV) As métricas obtidas com os melhores hiperparâmetros são mostradas. O fluxograma na Figura 6 pode ajudar a entender como o código funciona.

O algoritmo de regressão funciona de maneira similar ao algoritmo de classificação com as seguintes diferenças:

- (I) A variável dependente mantém seu formato numérico original, como anteriormente mencionado;

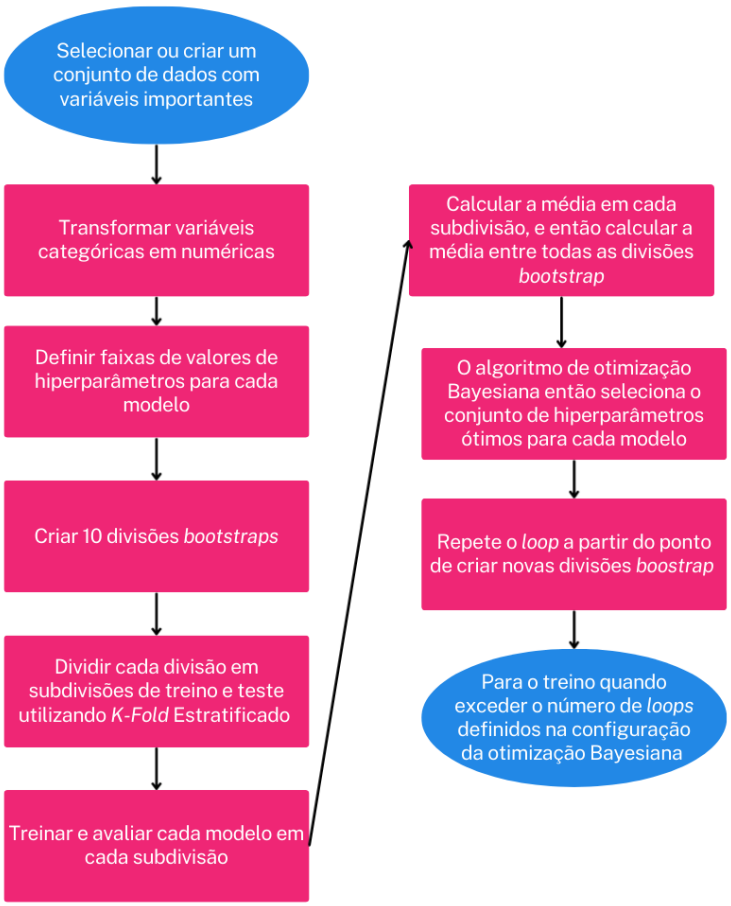

Figura 6 – Fluxograma do algoritmo de classificação.

- (II) Usa validação cruzada em cada conjunto *bootstrap* (não utiliza validação cruzada estratificada);
- (III) Treina e avalia o modelo selecionado (sendo os modelos definidos pelas funções *RandomForestRegressor*, *CatBoostRegressor* e *XGBRegressor* das bibliotecas *Scikit-learn*, *CatBoost* e *XGBoost*, respectivamente) em cada *fold*.

Alguns dados obtidos de pesquisas científicas mais recentes (a partir de 2022) foram adicionadas ao conjunto de dados original e toda a análise foi realizada novamente para checar o poder de generalização dos modelos criados de forma mais abrangente e robusta.

4 RESULTADOS E DISCUSSÕES

4.1 RESULTADOS NA PREVISÃO E OTIMIZAÇÃO DE GÁS DE SÍNTESE

Os algoritmos de RF e GBDT foram aplicados para prever a distribuição dos produtos finais (isto é, biocarvão, alcatrão e gás de síntese) em processos utilizando biomassa como matéria-prima. Os hiperparâmetros ótimos utilizados em cada um dos algoritmos foram obtidos após 50 iterações (testando combinações diferentes) usando a validação cruzada 10-*fold* com otimização bayesiana no conjunto de treino. O valor do melhor R^2 dos conjuntos de treino e teste para os três produtos são mostrados na Tabela 2. Com estes resultados é possível dizer que o algoritmo GBDT tem uma boa capacidade de previsão para esse conjunto de dados, visto que os valores do coeficiente de determinação R^2 foram altos e próximos nos conjuntos de treino e teste para os três produtos, indicando que *overfitting* foi efetivamente combatido durante o desenvolvimento do modelo de ML, enquanto que o algoritmo RF teve uma boa performance no treino, a performance nos conjuntos de teste foi baixa, indicando que o modelo sofre de alto índice de *overfitting*. Sendo assim, prova-se o motivo da escolha do GBDT para o novo modelo para prever a composição do gás de síntese.

Tabela 2 – Performance do treino e teste

Algoritmo de ML	Índice de performance	Biocarvão	Alcatrão	Gás de síntese
RF	Treino R^2	0.98	0.98	0.96
	Teste R^2	0.75	0.72	0.85
GBDT	Treino R^2	0.98	0.97	0.98
	Teste R^2	0.96	0.98	0.96

Em adição à previsão dos produtos resultantes da gaseificação, é importante identificar os fatores-chave que afetariam o rendimento do produto e descobrir suas correlações com os outros produtos de forma a melhorar a performance da gaseificação. Baseado na modelagem através de dados, pode-se convenientemente conseguir tais objetivos utilizando os modelos GBDT desenvolvidos a partir da interpretabilidade entre as entradas e as saídas (Li et al., 2023). As Figuras 7, 8 e 9 apresentam a importância das variáveis de entradas nos produtos de saída (biocarvão, alcatrão e gás de síntese). Para a predição do biocarvão, os teores de cinzas e hidrogênio, e a temperatura foram as três características mais importantes, podendo ser observado tais resultados na Figura 7, bem como a dependência parcial do alcatrão e do gás de síntese nas Figuras 11 e 12. Analisando-se a Figura 12, pode-se concluir que para um maior rendimento do gás de síntese, são mais adequadas as seguintes características: um baixo percentual do teor de nitrogênio em conjunto com um baixo percentual do teor de oxigênio entre 0% e 5% ou um alto teor de oxigênio entre 45% e 50% (ou seja, o teor de oxigênio deve-se manter 0 e 5% ou entre 45% e 50% para o maior rendimento de gás de síntese), juntamente de altas temperaturas (entre 800°C e 1000°C), o mesmo tipo de análise pode ser feito para os gráficos de dependência parcial

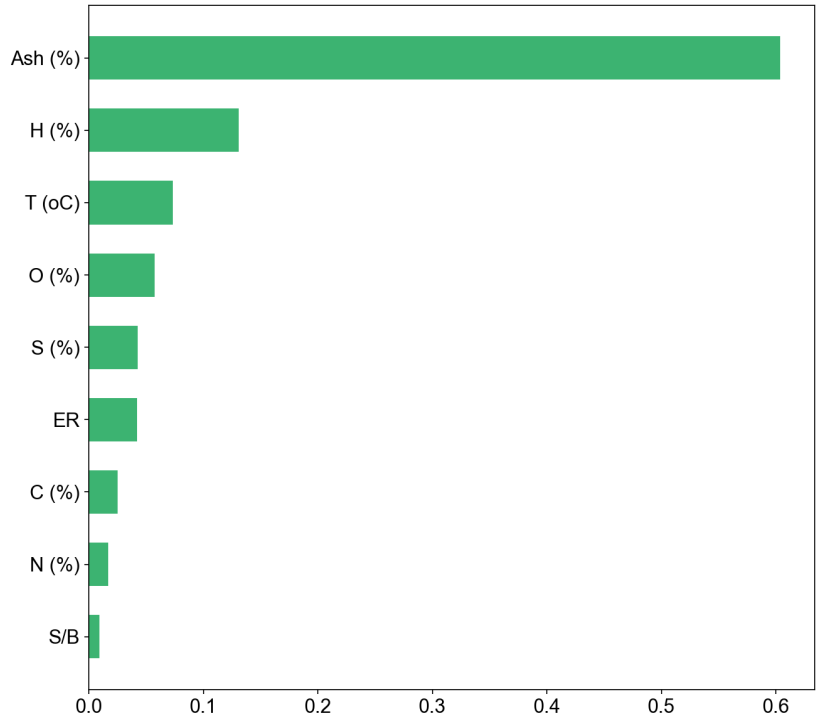

Figura 7 – Importância característica do biocarvão (autoria própria).

do biocarvão e do alcatrão. É interessante notar que o teor de nitrogênio é o fator mais importante para a predição de gás de síntese, e tem um impacto altamente negativo no percentual de gás de síntese. Um possível motivo para isso é que um alto teor de nitrogênio promoveria formação de alcatrão. Tem sido observado através de estudos experimentais que altas temperaturas estimulam a gaseificação do biocarvão e quebra térmica do alcatrão para produzir mais gás de síntese (Li et al., 2023).

Quanto ao alcatrão, ele tem um impacto negativo no rendimento do gás de síntese, que o faz um subproduto indesejado no processo de gaseificação. Principalmente na gaseificação auxiliada por catalisadores, alcatrão pode reduzir a atividade e tempo de vida de catalisadores ao se prender à superfície dos catalisadores. Portanto, é necessário investigar em detalhes como as variáveis de entrada afetam a produção de alcatrão. Observando a Figura 11, nota-se que um alto teor de nitrogênio, por exemplo, produz um maior rendimento de alcatrão, o que explica o motivo que o alto teor de nitrogênio diminui a proporção do gás de síntese, como comentado no parágrafo anterior. Combinando estes resultados comentados da Figura 12 com os resultados que não foram comentados (mas que podem ser analisados da mesma forma que foi feita para o gás de síntese) das Figuras 10 e 11, um baixo teor de nitrogênio ($< 5\%$) na biomassa e uma alta temperatura ($> 800°C$) são recomendados para um maior rendimento de gás de síntese e um menor rendimento de alcatrão. Além do mais, um baixo teor de cinzas na biomassa é mais benéfico do que teor nulo de cinzas para redução da geração de alcatrão, enquanto um maior aumento no teor de cinzas não teve óbvio impacto na produção de alcatrão, como pode ser observado na Figura 11. Em termos das características ER e S/B, elas tiveram impactos opostos

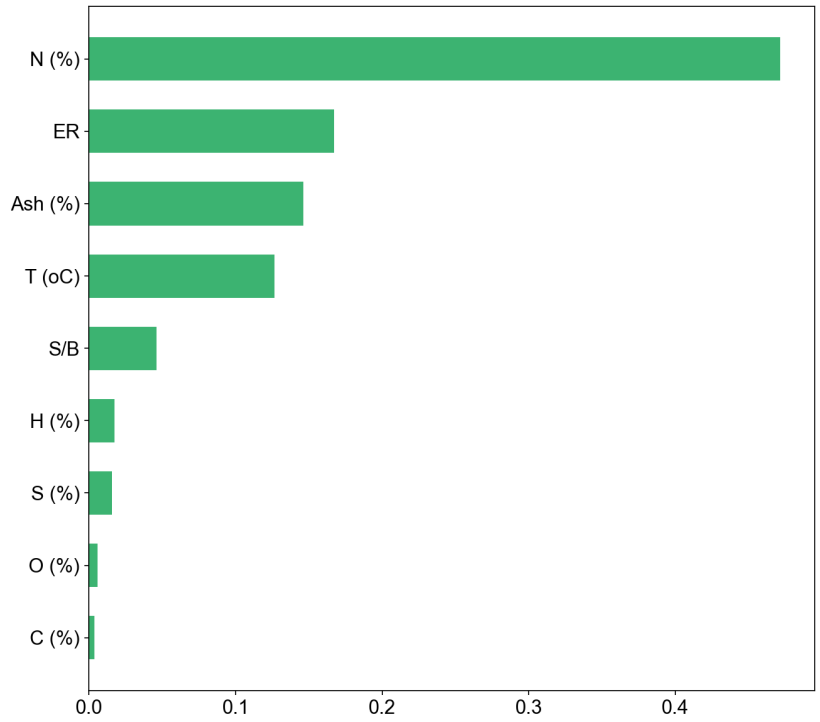

Figura 8 – Importância característica do alcatrão (autoria própria).

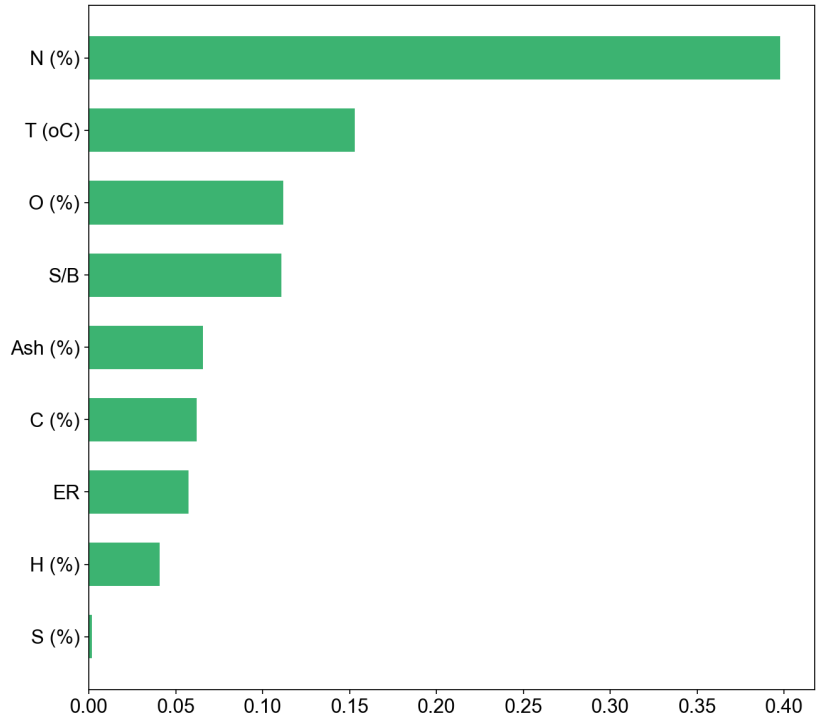

Figura 9 – Importância característica do gás de síntese (autoria própria).

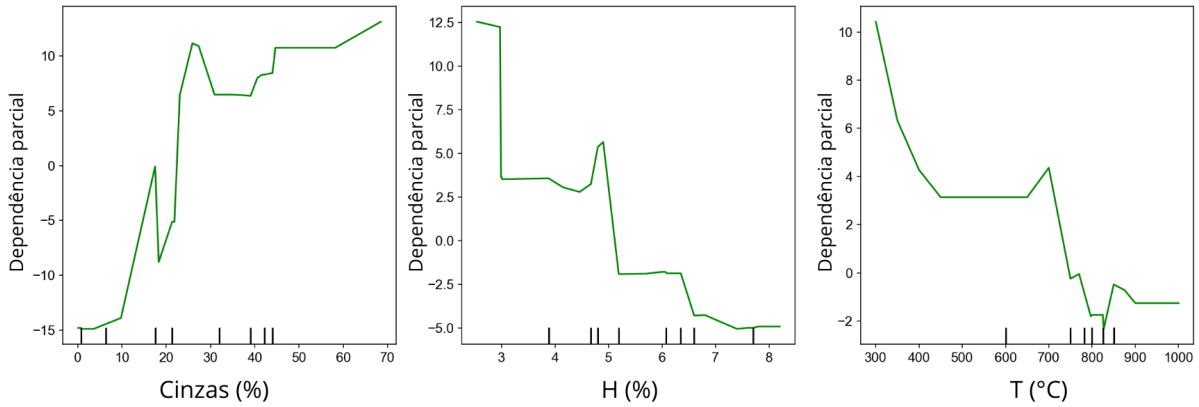

Figura 10 – Dependência parcial das três características mais importantes do biocarvão (autoria própria).

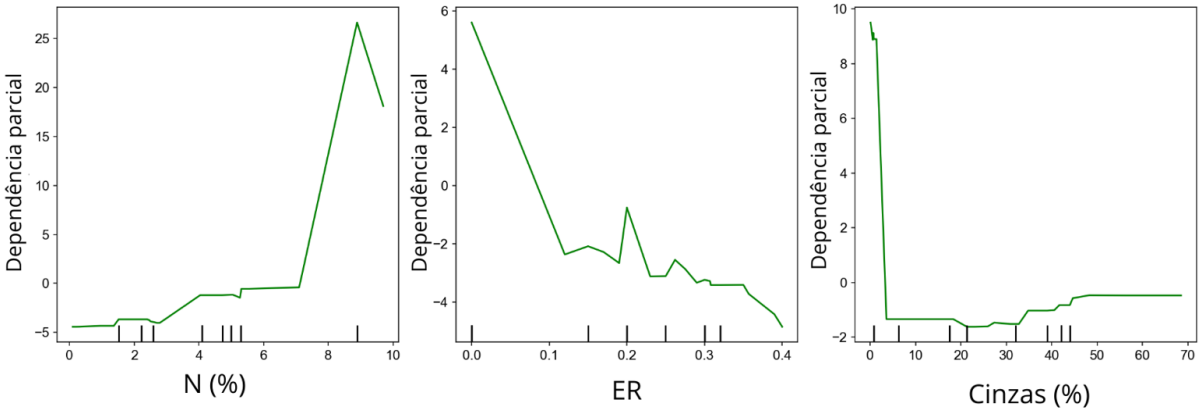

Figura 11 – Dependência parcial das três características mais importantes do alcatrão (autoria própria).

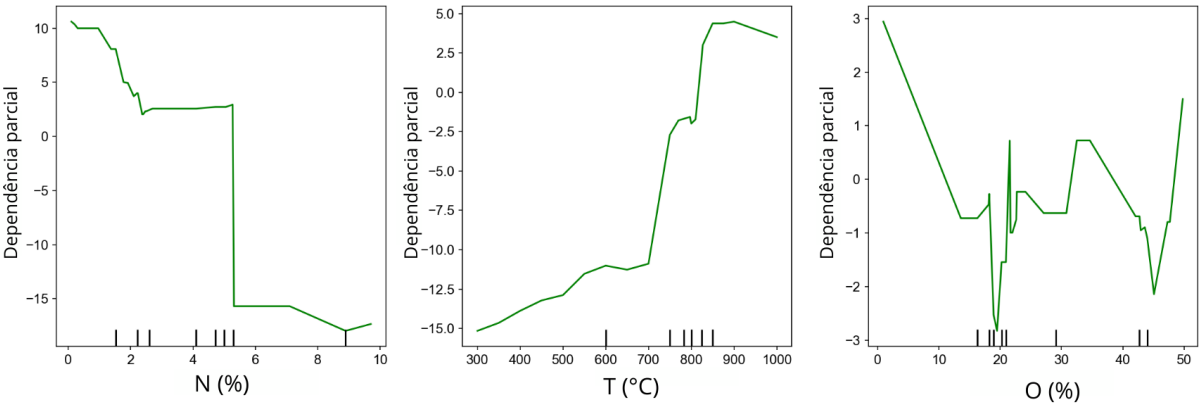

Figura 12 – Dependência parcial das três características mais importantes do gás de síntese (autoria própria).

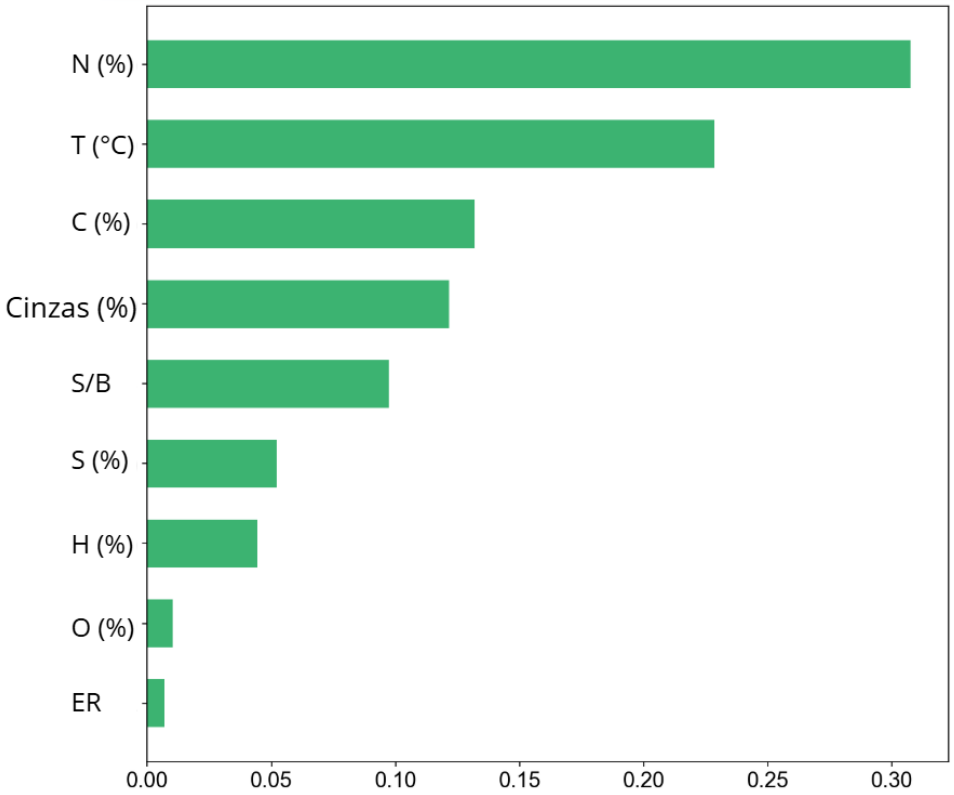

Figura 13 – Importância característica do componente H_2 do gás de síntese (autoria própria).

na produção de alcatrão, onde alto ER e baixo S/B são recomendados para evitar maior produção de alcatrão (Li et al., 2023).

Gás de síntese é o principal produto do processo de gaseificação que é uma mistura contendo H_2, CH_4, CO_2 e CO. Compreender a importância e relevância das variáveis de entrada para estes gases é útil para preparar um gás de síntese rico em H_2 através de ajustes da composição da biomassa e condições operacionais. A Figura 13 mostra que teor de nitrogênio, temperatura e teor de carbono foram as três variáveis mais importantes para o rendimento de H_2 no gás de síntese, seguidos pelo teor de cinzas, S/B e teor de enxofre, indicando que tanto as condições de gaseificação quanto as propriedades da biomassa têm papéis importantes no rendimento do H_2. As Figuras 14, 15 e 16 mostram as características mais importantes para o CH_4, CO_2 e CO, respectivamente. Há estudos que indicam que o teor de nitrogênio na biomassa pode ser convertido em amônia, cianeto de hidrogênio e óxidos de nitrogênio, que podem gerar impactos negativos na geração de H_2 e na qualidade do gás de síntese. Manyà et al. investigou o efeito das condições de operações na composição do gás de síntese e sugeriu que o rendimento de H_2 aumenta com o aumento da temperatura de gaseificação (Li et al., 2023).

As Figuras 17, 18, 19 e 20 indicam que uma alta temperatura (> 800°C) com alto S/B (> 2,0) e baixo ER (< 1,0) poderia melhorar a geração de H_2. Sob as temperaturas ótimas acima de 800°C, altos rendimentos de H_2 podem ser obtidos com baixos teores de cinzas e nitrogênio (entre 1%-5% e 0%-0,5%, respectivamente). Além do mais, resíduos de biomassa com alto teor de carbono (> 57%), alto teor de enxofre (> 1,5%), baixo teor de hidrogênio (< 4%) e alto teor

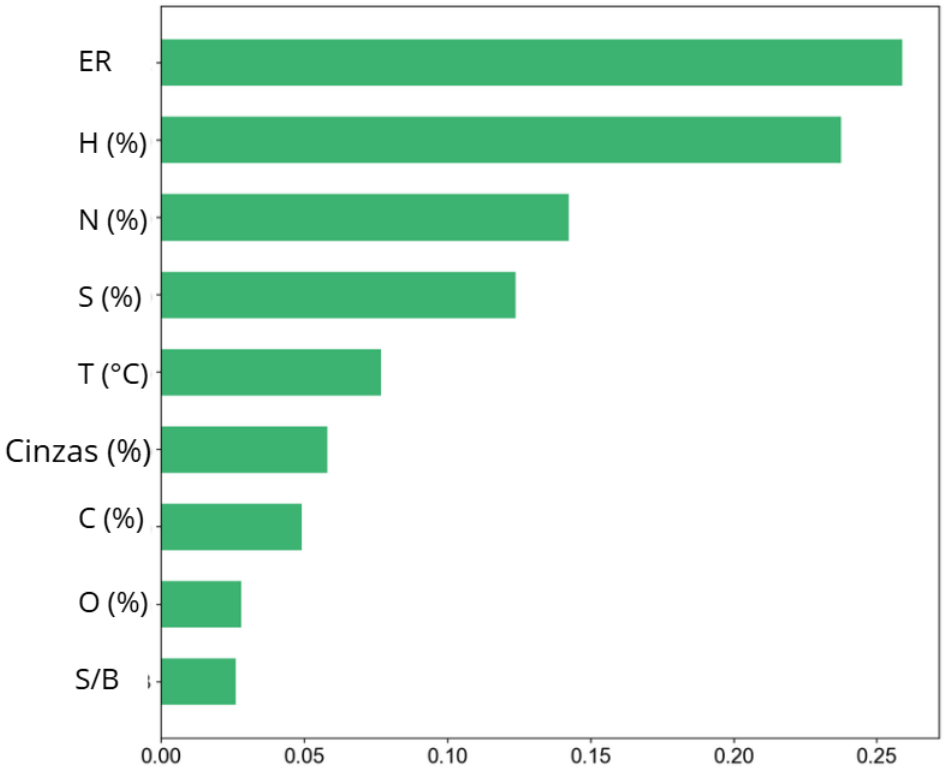

Figura 14 – Importância característica do componente CH_4 do gás de síntese (autoria própria).

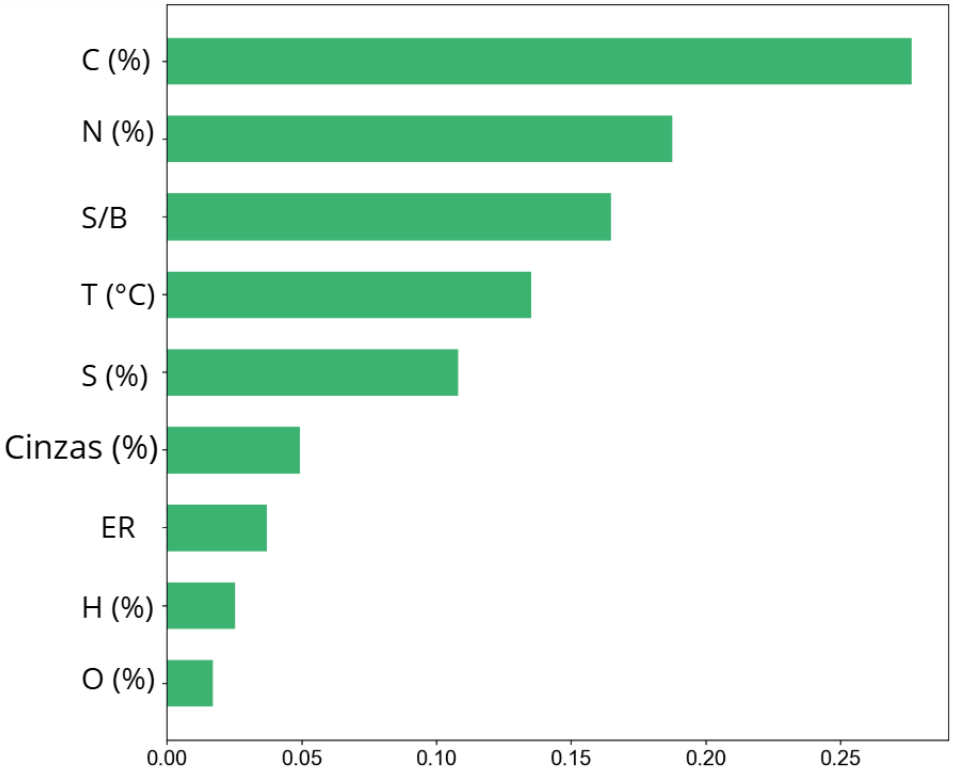

Figura 15 – Importância característica do componente CO_2 do gás de síntese (autoria própria).

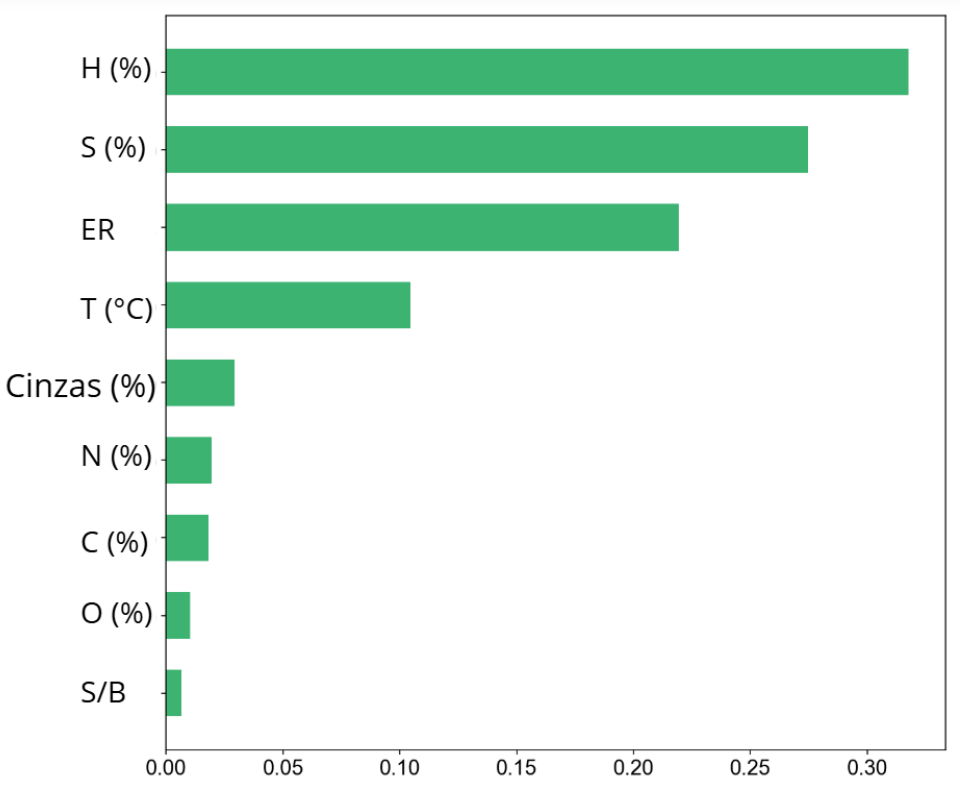

Figura 16 – Importância característica do componente CO do gás de síntese (autoria própria).

de oxigênio (> 20%) são recomendados para produção de gás de síntese. Fazendo uma análise similar para a composição de CO_2 que é o produto mais indesejado na composição do gás de síntese, obtêm-se os limites mínimos e máximos de cada uma destas variáveis a fim de otimizar a produção de gás de síntese com alto percentual de H_2. Utilizando tais limites juntamente do algoritmo de otimização por enxame de partículas encontra-se o seguinte resultado para a composição do gás de síntese (H_2, CO_2, CH_4 e CO): 45,41084767, 4,07190728, 36,47295398 e 15,08214417 mol/kg, respectivamente.

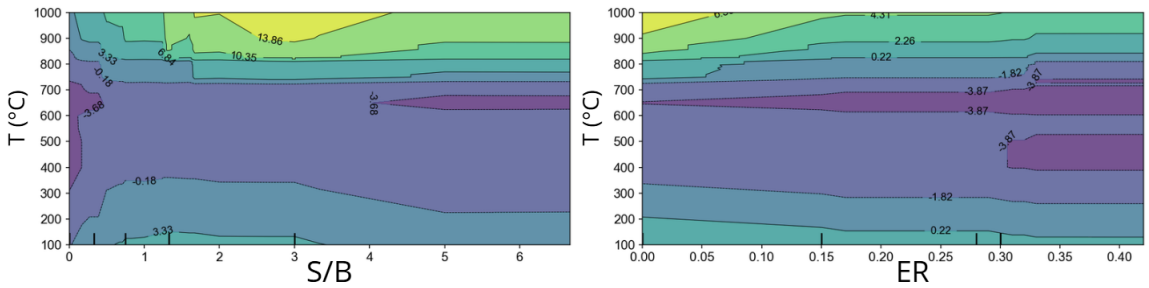

Figura 17 – Impactos da interação da temperatura com fatores S/B e ER no rendimento de H_2 no gás de síntese.

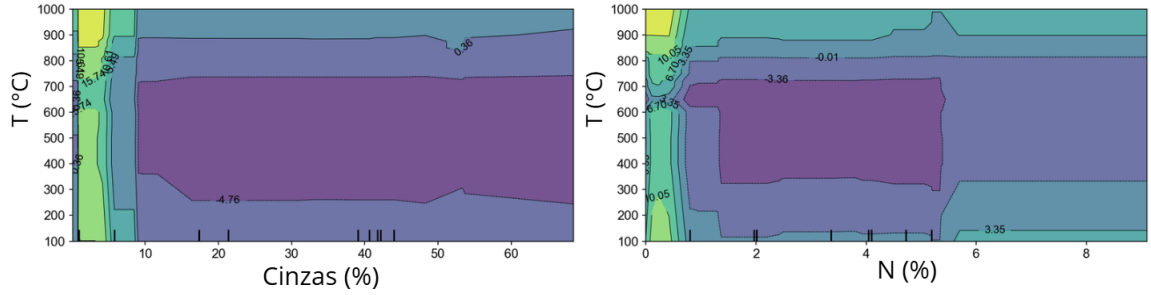

Figura 18 – Impactos da interação da temperatura com fatores Cinzas e N no rendimento de H_2 no gás de síntese.

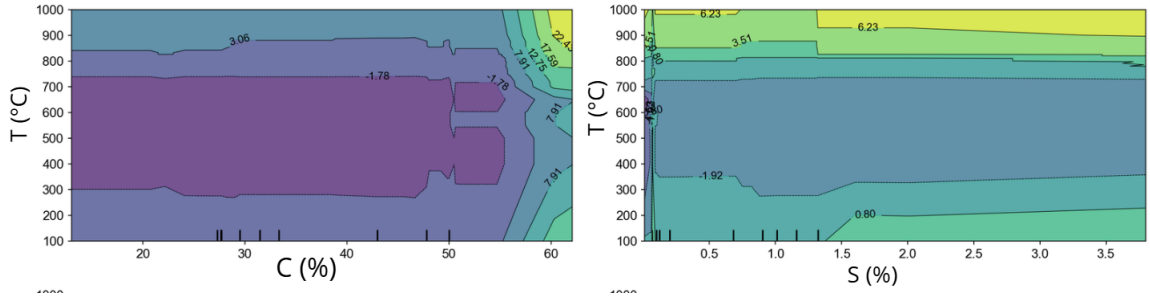

Figura 19 – Impactos da interação da temperatura com fatores C e S no rendimento de H_2 no gás de síntese.

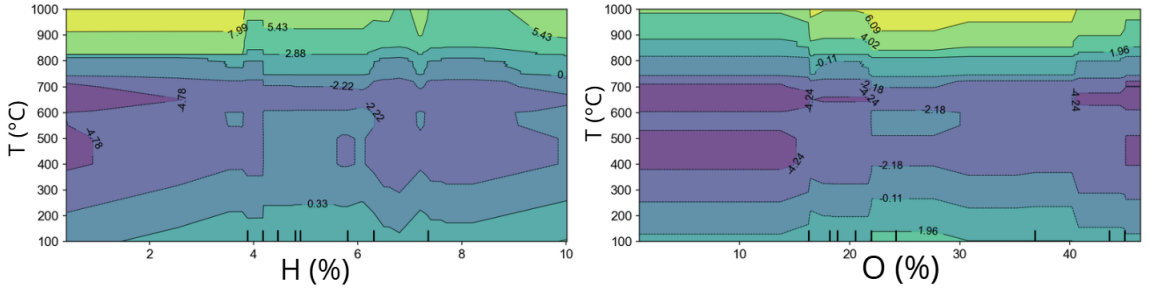

Figura 20 – Impactos da interação da temperatura com fatores H e O no rendimento de H_2 no gás de síntese.

4.2 RESULTADOS NA PREVISÃO DE CAPACITÂNCIA

As primeiras análises neste conjunto de dados foram as correlações de Spearman e Pearson (o mapa de calor com a correlação de Spearman pode ser visto na Figura 5), e infelizmente, no conjunto de dados original, a única variável com fraca correlação (valor > 0.3 (Mukaka, 2012)) com a variável dependente (capacitância específica) usando ambos os coeficientes de Spearman e de Pearson, foi a variável que indica se o material é um compósito ou não. Por conta dessa variável ser binária, uma interpretação sobre o que este valor significa é mais complicada. Usando coeficiente de Pearson, o valor apresentado pode ser interpretado como um tamanho do efeito que a diferença entre os dois grupos possui (Kornbrot, 2014) (isto é, representa quanto a média da variável contínua (capacitância específica) difere entre cada classe representada pela

variável binária), e como o conceito de correlação é sem significado se os dados são categóricos (exceto usando métodos como ponto bisserial, útil para variáveis dicotômicas, por exemplo). Mais criticamente, o coeficiente de Spearman funciona no valor ranqueado dos dados, mas se uma das variáveis é binária, não é recomendado utilizá-lo (King; Eckersley, 2019). Portanto, nenhuma informação útil pode ser obtida das correlações utilizadas, e consequentemente, todas as variáveis foram utilizadas para treinar o modelo.

Com os modelos de classificação treinados (usando 100% do conjunto de dados), os resultados obtidos são mostrados na Tabela 3

Tabela 3 – Média das métricas dos algoritmos de classificação

Nome do modelo	**Precisão (média)**	**Kappa (média)**	**MAE (média)**	**AUROC (média)**	**TPR (média)**
Random Forest	0.86	0.79	0.17	0.96	0.83
XGBoost	0.86	0.79	0.17	0.95	0.81

Observando apenas os valores da precisão, MAE e TPR, pode-se concluir que o modelo treino apresenta excelente capacidade de classificação para ambos os modelos. Ao remover as linhas de dados categorizadas com o número 13 na coluna de substratos/*substrate* (onde "13" representa que o substrato utilizado não foi reportado nos artigos, i.e., seriam dados nulos se não tivesse sido classificado como "13"), as métricas passaram por mudanças ínfimas. Removendo todas as linhas contendo valores nulos do conjunto de dados e treinando o modelo novamente, obtém-se as métricas mostradas na Tabela 4.

Tabela 4 – Média das métricas dos algoritmos de classificação (sem instâncias com valores nulos)

Nome do modelo	**Precisão (média)**	**Kappa (média)**	**MAE (média)**	**AUROC (média)**	**TPR (média)**
Random Forest	0.90	0.84	0.10	0.96	0.85
XGBoost	0.90	0.84	0.11	0.94	0.83

Embora precisão e TPR tenham aumentado e MAE diminuiu, o poder de generalização do modelo treinado foi enormemente reduzido, dado que haviam 226 linhas de dados nulos, representando 38,89% do conjunto de dados original. Esses testes podem ajudar a entender como mudanças que podem parecer positivas, podem ter um forte impacto negativo, dependendo do objetivo do modelo treinado. Se o objetivo é apenas precisão ao analisar novos valores próximos dos dados existentes, reduzir o tamanho do conjunto de dados ao custo de deletar linhas nulas pode ser viável, especialmente se o conjunto de dados é representativo o suficiente.

Em seguida, 20 linhas de dados (a partir do ano de 2022) foram adicionadas para o conjunto de dados, obtendo os resultados mostrados na Tabela 5. Comparando as Tabelas 3 e 5,

observa-se que adicionar 20 linhas de dados não causa mudanças significativas nas métricas do modelo.

Tabela 5 – Média das métricas dos algoritmos de classificação (com 20 pontos de dados adicionais)

Nome do modelo	**Precisão (média)**	**Kappa (média)**	**MAE (média)**	**AUROC (média)**	**TPR (média)**
Random Forest	0.85	0.78	0.19	0.95	0.81
XGBoost	0.85	0.77	0.19	0.94	0.80

Agora, utilizando algoritmos de regressão e usando novamente o conjunto de dados original obtém-se as métricas para cada um dos modelos mostradas na Tabela 6:

Tabela 6 – Média das métricas dos algoritmos de regressão.

Nome do modelo	R^2 **(média)**	**MAE (média)**	**RMSE (média)**	**MAPE (média)**
Random Forest	0.70	96.98	190.36	0.43
XGBoost	0.77	83.07	166.03	0.36
CatBoost	0.80	68.65	151.85	0.31

Analisando os resultados obtidos neste treino, *CatBoost* seria o modelo mais apropriado (para este conjunto de dados) porque ele obteve o melhor R^2, e relativamente baixos MAE e MAPE (quando comparado com os outros 2 modelos). Um cuidado especial deve ser tomado com RMSE e MAPE por conta de serem métricas sensíveis a *outliers* (Davydenko; Fildes, 2016), (Hyndman; Koehler, 2006). O estudo Chicco, Warrens e Jurman (2021) indica que métricas como MAE, MAPE ou RMSE não ajudam em saber a qualidade da performance do modelo de regressão utilizando apenas seus valores individuais, portanto, será priorizada a métrica R^2 nesta seção para os algoritmos de regressão. O ranking de importância das características (do inglês, *feature importance*, pode ser interpretada como a quantidade de poder preditivo que uma característica possui (Covert; Lundberg; Lee, 2020)) incluindo apenas as 4 variáveis mais importantes para cada modelo é mostrado na Tabela 7, notando que as características mais importantes variam enormemente dependendo do modelo:

Então, os 20 pontos de dados adicionados foram adicionados para o modelo de regressão e os resultados obtidos para os modelos treinados são mostrados na Tabela 8.

Observando os resultados da Tabela 8, a degradação do modelo treinado é evidente quando adiciona-se os mesmos 20 pontos de dados adicionados durante a etapa de classificação. Esta degradação, aparentemente, é devido a valores extremamente grandes dos dados adicionados que alteram a média, o desvio padrão (std - *standard deviation*) e o Coeficiente de Variação (CoVa) do conjunto de dados como pode ser visto na Tabela 9 (dos conjuntos de dados original e

Tabela 7 – Importância das características dos algoritmos de regressão.

Nome do modelo	**1ª característica**	**2ª característica**	**3ª característica**	**4ª característica**
Random Forest	Janela de potencial	Compósito ou não	Densidade de corrente	Área superficial
XGBoost	C (binário)	Compósito ou não	T (binário)	Janela de potencial
CatBoost	Janela de potencial	Código da morfologia	Compósito ou não	Área superficial

Nome do modelo	R^2 **(média)**	**MAE (média)**	**RMSE (média)**	**MAPE (média)**
Random Forest	0.49	121.52	303.41	0.49
XGBoost	0.61	94.79	269.09	0.37
CatBoost	0.61	122.45	277.73	0.52

Tabela 8 – Média das métricas dos algoritmos de regressão (com 20 pontos de dados adicionados).

modificado), onde CoVa é uma medida de propagação para um conjunto de dados definida como 100*std/média (Goodier, 2011):

Características do conjunto de dados	**Área superficial (m2/g)**	**Janela de potencial (V)**	**Densidade de corrente (A/g)**	**Capacitância específica (F/g)**
Média - Conjunto	102.96	0.86	6.03	357.01
Std - Conjunto	154.47	0.50	10.15	349.11
Std/Média (%) (Original)	150%	58%	168%	97%
Média - Conjunto	115.21	0.88	5.90	381.439
Std - Conjunto	216.75	0.52	10.03	439.09
Std/Média (%) (Modificado)	188%	59%	170%	115%

Tabela 9 – Média, desvio padrão e coeficiente de variação dos conjuntos de dados.

Como pode ser visto na Tabela 9, o CoVa da área superficial e da capacitância específica aumentou em 38% e 18%, respectivamente, comparando o conjunto de dados modificado com o original, indicando uma maior variação destas duas variáveis e implicando uma menor representatividade do conjunto de dados modificado. Confirmando o que foi anteriormente comentado que um conjunto de dados maior não implicará em resultados melhores ao utilizá-lo com algoritmos de aprendizagem de máquina, tal que a representatividade do conjunto de dados é um fator crucial. Para uma melhor compreensão, pode-se observar os maiores valores para essas duas variáveis na Tabela 10. Serão mostrados apenas os 4 maiores valores das duas variáveis para

se ter uma noção de como os 20 pontos de dados adicionados impactaram tanto a performance do modelo (prestando atenção aos *outliers* incluídos).

Valores do conjunto de dados	**1° valor**	**2° valor**	**3° valor**	**4° valor**
Área superficial (m2/g)	809.00	809.00	809.00	809.00
Capacitância específica (F/g) (Original)	2155.60	1980.92	1980.00	1978.00
Área superficial (m2/g)	3005.00	809.00	809.00	809.00
Capacitância específica (F/g) (Modificado)	5157.81	3030.54	2921.80	2648.80

Tabela 10 – Maiores valores da área superficial e capacitância específica nos conjuntos de dados.

O motivo da performance dos algoritmos de classificação não deteriorar da mesma forma que os modelos de regressão é provavelmente por conta dos algoritmos de classificação operarem em faixas de valores, ou seja, por conta dos 20 pontos de dados não serem uma grande quantidade em um conjunto de dados que já possui 581 amostras, a faixa superior de valores (se todos os 20 valores fossem da faixa superior) não seria modificada (considerando que antes disso não houvesse um enorme desbalanceamento de classes).

Após treinar os modelos de classificação e regressão, os modelos foram utilizados para prever a faixa de classificação e o valor da capacitância específica do material apresentado no estudo Yesilbag et al. (2022) (esse material não foi incluído no conjunto de dados utilizado para treinar os modelos), que possui uma capacitância específica de 690 F/g, para fazer esse tipo de previsão, a função *predict* de cada modelo foi utilizada, as características do material (valor de cada variável) foi utilizada. Utilizando o modelo de classificação (com o conjunto de dados original), tanto o RF quanto o XGBoost classificaram a amostra pertencendo à classe C (classe C representa valores de 500 a 1000 F/g), sendo a probabilidade de 69% com RF e 95% com XGBoost. Usando o modelo de regressão (com o conjunto original), as previsões do RF, XGBoost e CatBoost resultaram em 450,46 F/g, 436,88 F/g e 315,61 F/g, respectivamente, indicando que os 3 modelos sofreram *overfitting*, mesmo que nenhum dos 3 modelos tenha obtido um alto R^2, sendo CatBoost o algoritmo que mais sofreu com *overfitting* entre os 3 (enquanto seu R^2 foi o melhor entre os 3 modelos). Portanto, os modelos de regressão foram utilizados com o conjunto de dados modificado (com 19 pontos adicionados, isto é, sem o ponto de dados que será previsto), os modelos treinados obtiveram R^2 de 0,48, 0,64 e 0,69, e as previsões resultaram em 692,28 F/g, 654,12 F/g e 392,66 F/g, para o RF, XGBoost e CatBoost, respectivamente. Em outras palavras, embora RF tenha obtido o pior R^2 com esse conjunto de dados, ele preveu o valor mais próximo ao valor que deveria ser previsto (tendo alta precisão, ao menos para este ponto específico, uma análise mais robusta em outros pontos de dados não pertencentes ao conjunto de treino deve ser feita para verificar se essa precisão se mantém ou se foi apenas sorte a precisão ser tão alta).

Através destes exemplos, fica mais fácil de observar como *overfitting* e generalização funcionam na previsão, e um estudo amplamente robusto deveria ser feito dependendo do objetivo ou de quais pontos de dados serão analisados com seu modelo.

5 CONCLUSÃO

Neste trabalho foram apresentadas duas abordagens utilizando algoritmos de aprendizagem de máquina, uma das abordagens sendo utilizada para a previsão de produtos finais em processo termoquímico que utiliza biomassa e outra abordagem para a previsão de capacitância de supercapacitores. Nas duas abordagens obteve-se resultados interessantes no que tange a compreensão de como montar modelos com resultados satisfatórios.

Ficou claro através das abordagens que o tratamento de dados antes de montar o modelo é essencial, seja na escolha de quais variáveis utilizar ou na remoção de amostras de dados com valores muito diferentes do resto do conjunto. A escolha de qual algoritmo de previsão utilizar sendo um dos pontos mais importantes no sucesso do modelo, como pode ser visto no caso da previsão de capacitância de supercapacitores, onde o modelo utilizando o algoritmo *Random Forest* obteve um coeficiente de determinação de 0,49 enquanto os modelos utilizando *XGBoost* e *CatBoost* obtiveram 0,61.

Os coeficientes de determinação obtidos na previsão de produtos finais para o conjunto de teste para o biocarvão, alcatrão e gás de síntese foram 0,96, 0,98 e 0,96, respectivamente, sendo que este resultado foi melhor do que o encontrado em Li et al. (2023), onde os coeficientes encontrados pelo autor foram 0,95, 0,96 e 0,91, respectivamente. O modelo criado para a previsão da composição do gás de síntese foi melhorado utilizando a otimização por enxame de partículas para se obter um maior índice de H_2 que é o componente desejável e redução dos gases CO_2, CH_4 e CO resultando na seguinte proporção para (H_2, CO_2, CH_4 e CO): 45,41084767, 4,07190728, 36,47295398 e 15,08214417 mol/kg, respectivamente, enquanto que os resultados encontrados em Li et al. (2023) foram 44,34, 40,72, 3,29 e 18,25 mol/kg, respectivamente, ou seja, a combinação dos valores das variáveis de entrada encontrada neste trabalho foram melhores que a encontrada em Li et al. (2023), provavelmente por conta do algoritmo de otimização por enxame de partículas utilizado aqui, enquanto que o trabalho Li et al. (2023) não utilizou nenhum algoritmo nesta etapa do processo de previsão. Outra diferença importante de ser citada é que no trabalho de Li et al. (2023), os pesquisadores obtiveram os rendimentos de CO_2 e CH_4 igual a 40,72 mol/kg e 3,29 mol/kg, respectivamente, sendo estes valores bem diferentes dos valores encontrados nesta dissertação, provavelmente foi um erro dos pesquisadores do artigo.

Diante do exposto, fica claro o sucesso em utilizar métodos de aprendizagem de máquina tanto na previsão de produtos finais de processos termoquímicos, quanto na previsão de capacitância de supercapacitores, bem como deve-se levar em conta diversos fatores ao montar os modelos de previsão a fim de se obter os melhores resultados, poupando tempo e recursos que seriam necessários para realizar inúmeros testes em laboratório.

5.1 PROPOSTAS PARA TRABALHOS FUTUROS

- Considerar submodelos por tipo de processo (pirólise/gaseificação, por exemplo) e/ou por classe de biomassa e/ou incorporar mais *features* típicas da pirólise (taxa de aquecimento,

tempo de residência de voláteis, etc);

- Testar outros algoritmos de aprendizagem de máquina que podem ser úteis para montar modelos nesses tipos de processos (como KNN, por exemplo) e/ou testar outros algoritmos de otimização de parâmetros (algoritmos MPSO, por exemplo);
- Criar um modelo que analise os valores das variáveis de entrada da biomassa e faça uma previsão de valores de propriedades se um supercapacitor fosse fabricado com o biocarvão gerado a partir de tal biomassa, ou seja, o modelo faria a previsão tanto das características do biocarvão quanto uma previsão das características de um supercapacitor fabricado utilizando-se esse biocarvão.

REFERÊNCIAS

AKHTAR, Ali; KREPL, Vladimir; IVANOVA, Tatiana. A combined overview of combustion, pyrolysis, and gasification of biomass. **Energy & Fuels**, ACS Publications, v. 32, n. 7, p. 7294–7318, 2018. Citado 2 vezes nas páginas 15 e 16.

ALKHARUSI, Hussain. Categorical variables in regression analysis: A comparison of dummy and effect coding. **International Journal of Education**, Macrothink Institute Inc., v. 4, n. 2, p. 202, 2012. Citado na página 21.

ALSABER, Ahmad R; PAN, Jiazhu; AL-HURBAN, Adeeba. Handling complex missing data using random forest approach for an air quality monitoring dataset: a case study of kuwait environmental data (2012 to 2018). **International Journal of Environmental Research and Public Health**, MDPI, v. 18, n. 3, p. 1333, 2021. Citado na página 61.

ALTHNIAN, Alhanoof et al. Impact of dataset size on classification performance: an empirical evaluation in the medical domain. **Applied sciences**, MDPI, v. 11, n. 2, p. 796, 2021. Citado na página 60.

ANGUITA, Davide et al. The'k'in k-fold cross validation. In: **Esann**. [S.l.: s.n.], 2012. v. 102, p. 441–446. Citado 2 vezes nas páginas 41 e 42.

BARBOZA, Flavio; KIMURA, Herbert; ALTMAN, Edward. Machine learning models and bankruptcy prediction. **Expert systems with applications**, Elsevier, v. 83, p. 405–417, 2017. Citado na página 21.

BATES, Stephen; HASTIE, Trevor; TIBSHIRANI, Robert. Cross-validation: what does it estimate and how well does it do it? **Journal of the American Statistical Association**, Taylor & Francis, v. 119, n. 546, p. 1434–1445, 2024. Citado na página 41.

BELAYNEH, A et al. Coupling machine learning methods with wavelet transforms and the bootstrap and boosting ensemble approaches for drought prediction. **Atmospheric research**, Elsevier, v. 172, p. 37–47, 2016. Citado 2 vezes nas páginas 39 e 40.

BERK, Richard A et al. **Statistical learning from a regression perspective**. [S.l.]: Springer, 2008. v. 14. Citado 10 vezes nas páginas 25, 26, 27, 30, 31, 32, 33, 34, 35 e 38.

BIAU, Gérard; , Erwan . A random forest guided tour. **Test**, Springer, v. 25, n. 2, p. 197–227, 2016. Citado 5 vezes nas páginas 43, 44, 45, 47 e 49.

BINIEK-POSKART, Anna et al. The application of lignocellulosic biomass waste in the iron and steel industry in the context of challenges related to the energy crisis. **Energies**, MDPI, v. 16, n. 18, p. 6662, 2023. Citado na página 14.

BOYD, Stephen P; VANDENBERGHE, Lieven. **Convex optimization**. [S.l.]: Cambridge university press, 2004. Citado na página 50.

BREIMAN, Leo. Bagging predictors. **Machine learning**, Springer, v. 24, p. 123–140, 1996. Citado 3 vezes nas páginas 39, 40 e 43.

BREIMAN, Leo. Arcing classifier (with discussion and a rejoinder by the author). **The annals of statistics**, Institute of Mathematical Statistics, v. 26, n. 3, p. 801–849, 1998. Citado 2 vezes nas páginas 39 e 40.

BREIMAN, Leo. Random forests. **Machine learning**, Springer, v. 45, p. 5–32, 2001. Citado 3 vezes nas páginas 43, 45 e 47.

BREIMAN, Leo et al. **Classification and regression trees**. [S.l.]: Routledge, 2017. Citado 8 vezes nas páginas 24, 30, 31, 35, 36, 37, 43 e 47.

BROWNE, Michael W. Cross-validation methods. **Journal of mathematical psychology**, Elsevier, v. 44, n. 1, p. 108–132, 2000. Citado na página 42.

CETESB. Ácido acético: Ficha de informação toxicológica. 2012. Divisão de Toxicologia Humana e Saúde Ambiental. Disponível em: https://cetesb.sp.gov.br/laboratorios/wp-content/uploads/sites/24/2020/07/A%CC%81cido-ace%CC%81tico.pdf. Acesso em: 15 mai. 2025. Citado na página 18.

CHAIBI, Mohamed et al. Machines learning models based on feature selection and bayesian optimization for predicting daily global solar radiation. **Lhoussaine and Berrada, Mohamed and El Hmaidi, Abdellah, Machines Learning Models Based on Feature Selection and Bayesian Optimization for Predicting Daily Global Solar Radiation**, 2021. Citado na página 58.

CHEN, Jing et al. Joint learning with both classification and regression models for age prediction. In: IOP PUBLISHING. **Journal of Physics: Conference Series**. [S.l.], 2019. v. 1168, n. 3, p. 032016. Citado na página 66.

CHEN, Tianqi; GUESTRIN, Carlos. Xgboost: A scalable tree boosting system. In: **Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining**. [S.l.: s.n.], 2016. p. 785–794. Citado 3 vezes nas páginas 7, 56 e 57.

CHICCO, Davide; JURMAN, Giuseppe. The matthews correlation coefficient (mcc) should replace the roc auc as the standard metric for assessing binary classification. **BioData Mining**, Springer, v. 16, n. 1, p. 4, 2023. Citado na página 61.

CHICCO, Davide; WARRENS, Matthijs J; JURMAN, Giuseppe. The coefficient of determination r-squared is more informative than smape, mae, mape, mse and rmse in regression analysis evaluation. **Peerj computer science**, PeerJ Inc., v. 7, p. e623, 2021. Citado na página 77.

CHOK, Nian Shong. **Pearson's versus Spearman's and Kendall's correlation coefficients for continuous data**. Tese (Doutorado) — University of Pittsburgh, 2010. Citado na página 65.

COUSINS, Cyrus; RIONDATO, Matteo. Cadet: interpretable parametric conditional density estimation with decision trees and forests. **Machine Learning**, Springer, v. 108, p. 1613–1634, 2019. Citado 2 vezes nas páginas 25 e 26.

COVERT, Ian; LUNDBERG, Scott M; LEE, Su-In. Understanding global feature contributions with additive importance measures. **Advances in neural information processing systems**, v. 33, p. 17212–17223, 2020. Citado na página 77.

CUMMINGS, Peter. Missing data and multiple imputation. **JAMA pediatrics**, v. 167, n. 7, 2013. Citado na página 61.

DAVYDENKO, Andrey; FILDES, Robert. Forecast error measures: critical review and practical recommendations. **Business Forecasting: Practical Problems and Solutions. John Wiley & Sons Inc**, 2016. Citado na página 77.

DEEBANSOK, Siraprapha et al. Capacitive tendency concept alongside supervised machine-learning toward classifying electrochemical behavior of battery and pseudocapacitor materials. **Nature Communications**, Nature Publishing Group UK London, v. 15, n. 1, p. 1133, 2024. Citado na página 20.

DIVINE, Douglas Chinenye et al. Enhancing biomass pyrolysis: Predictive insights from process simulation integrated with interpretable machine learning models. **Fuel**, Elsevier, v. 366, p. 131346, 2024. Citado na página 64.

DOMINGOS, Pedro; PAZZANI, Michael. On the optimality of the simple bayesian classifier under zero-one loss. **Machine learning**, Springer, v. 29, p. 103–130, 1997. Citado na página 28.

DONG, Zixun et al. Machine learning prediction of pyrolytic products of lignocellulosic biomass based on physicochemical characteristics and pyrolysis conditions. **Bioresource Technology**, Elsevier, v. 367, p. 128182, 2023. Citado 2 vezes nas páginas 17 e 63.

DOROGUSH, Anna Veronika; ERSHOV, Vasily; GULIN, Andrey. Catboost: gradient boosting with categorical features support. **arXiv preprint arXiv:1810.11363**, 2018. Citado na página 57.

EMMANUEL, Tlamelo et al. A survey on missing data in machine learning. **Journal of Big data**, Springer, v. 8, n. 1, p. 140, 2021. Citado 2 vezes nas páginas 60 e 61.

FAHMY, Tamer YA et al. Biomass pyrolysis: past, present, and future. **Environment, Development and Sustainability**, Springer, v. 22, p. 17–32, 2020. Citado na página 15.

FIELD, Christopher B; CAMPBELL, J Elliott; LOBELL, David B. Biomass energy: the scale of the potential resource. **Trends in ecology & evolution**, Elsevier, v. 23, n. 2, p. 65–72, 2008. Citado na página 14.

FIGUEROA, Rosa L et al. Predicting sample size required for classification performance. **BMC medical informatics and decision making**, Springer, v. 12, n. 1, p. 8, 2012. Citado na página 60.

FOODY, Giles M. Explaining the unsuitability of the kappa coefficient in the assessment and comparison of the accuracy of thematic maps obtained by image classification. **Remote sensing of environment**, Elsevier, v. 239, p. 111630, 2020. Citado na página 62.

FRAZIER, Peter I. A tutorial on bayesian optimization. **arXiv preprint arXiv:1807.02811**, 2018. Citado na página 58.

FREUND, Yoav; SCHAPIRE, Robert E. A decision-theoretic generalization of on-line learning and an application to boosting. **Journal of computer and system sciences**, Elsevier, v. 55, n. 1, p. 119–139, 1997. Citado na página 40.

FRIEDMAN, Jerome H. Greedy function approximation: a gradient boosting machine. **Annals of statistics**, JSTOR, p. 1189–1232, 2001. Citado 8 vezes nas páginas 40, 42, 50, 51, 52, 53, 54 e 55.

GAJDZIK, Bożena et al. The influence of the global energy crisis on energy efficiency: A comprehensive analysis. **Energies**, MDPI, v. 17, n. 4, p. 947, 2024. Citado na página 12.

GHOJOGH, Benyamin; CROWLEY, Mark. The theory behind overfitting, cross validation, regularization, bagging, and boosting: tutorial. **arXiv preprint arXiv:1905.12787**, 2019. Citado na página 40.

GHOSH, Sourav; RAO, G Ranga; THOMAS, Tiju. Machine learning-based prediction of supercapacitor performance for a novel electrode material: Cerium oxynitride. **Energy Storage Materials**, Elsevier, v. 40, p. 426–438, 2021. Citado na página 64.

GOODIER, John. The cambridge dictionary of statistics. **Reference Reviews**, Emerald Group Publishing Limited, v. 25, n. 5, p. 37–38, 2011. Citado na página 78.

GRANDINI, Margherita; BAGLI, Enrico; VISANI, Giorgio. Metrics for multi-class classification: an overview. **arXiv preprint arXiv:2008.05756**, 2020. Citado na página 62.

GUYON, Isabelle; ELISSEEFF, André. An introduction to variable and feature selection. **Journal of machine learning research**, v. 3, n. Mar, p. 1157–1182, 2003. Citado na página 60.

GYÖRFI, László et al. **A distribution-free theory of nonparametric regression**. [S.l.]: Springer, 2002. Citado na página 49.

HALL, David O. Biomass energy. **Energy policy**, Elsevier, v. 19, n. 8, p. 711–737, 1991. Citado na página 14.

HASTIE, Trevor et al. **The elements of statistical learning: data mining, inference, and prediction**. [S.l.]: Springer, 2009. v. 2. Citado 2 vezes nas páginas 41 e 42.

HENDRIKS, ATWM; ZEEMAN, Grietje. Pretreatments to enhance the digestibility of lignocellulosic biomass. **Bioresource technology**, Elsevier, v. 100, n. 1, p. 10–18, 2009. Citado na página 14.

HYNDMAN, Rob J; KOEHLER, Anne B. Another look at measures of forecast accuracy. **International journal of forecasting**, Elsevier, v. 22, n. 4, p. 679–688, 2006. Citado na página 77.

IRO, Zaharaddeen S; SUBRAMANI, C; DASH, SS. A brief review on electrode materials for supercapacitor. **International Journal of Electrochemical Science**, Elsevier, v. 11, n. 12, p. 10628–10643, 2016. Citado 2 vezes nas páginas 12 e 18.

ISHAQ, Muhammad et al. Machine learning based missing data imputation in categorical datasets. **IEEE Access**, IEEE, v. 12, p. 88332–88344, 2024. Citado na página 60.

JAIN, NK; NANGIA, Uma; JAIN, Jyoti. A review of particle swarm optimization. **Journal of The Institution of Engineers (India): Series B**, Springer, v. 99, n. 4, p. 407–411, 2018. Citado na página 59.

JHA, Swarn et al. Machine learning-assisted materials development and device management in batteries and supercapacitors: performance comparison and challenges. **Journal of Materials Chemistry A**, Royal Society of Chemistry, v. 11, n. 8, p. 3904–3936, 2023. Citado na página 13.

JITAPUNKUL, Kulpavee et al. Insights into heteroatom-doped graphene supercapacitor data through manual data separation and statistical analysis. **The Journal of Physical Chemistry C**, ACS Publications, v. 127, n. 37, p. 18316–18326, 2023. Citado na página 20.

KAN, Tao; STREZOV, Vladimir; EVANS, Tim J. Lignocellulosic biomass pyrolysis: A review of product properties and effects of pyrolysis parameters. **Renewable and sustainable energy reviews**, Elsevier, v. 57, p. 1126–1140, 2016. Citado 3 vezes nas páginas 14, 15 e 16.

KHAN, Suliman et al. Biodiesel production from algae to overcome the energy crisis. **HAYATI Journal of Biosciences**, Elsevier, v. 24, n. 4, p. 163–167, 2017. Citado na página 14.

KHAN, Shahidul Islam; HOQUE, Abu Sayed Md Latiful. Sice: an improved missing data imputation technique. **Journal of big Data**, Springer, v. 7, n. 1, p. 37, 2020. Citado na página 61.

KHEDULKAR, Akhil Pradiprao et al. Agricultural waste to real worth biochar as a sustainable material for supercapacitor. **Science of the Total Environment**, Elsevier, v. 869, p. 161441, 2023. Citado na página 19.

KING, Andrew P; ECKERSLEY, Robert. **Statistics for biomedical engineers and scientists: how to visualize and analyze data**. [S.l.]: Academic Press, 2019. Citado na página 76.

KITTLER, Josef; ROLI, Fabio. Multiple classifier systems. **Lecture notes in computer science**, Springer, v. 1857, 2000. Citado na página 40.

KLEIN, Aaron et al. Fast bayesian optimization of machine learning hyperparameters on large datasets. In: PMLR. **Artificial intelligence and statistics**. [S.l.], 2017. p. 528–536. Citado 2 vezes nas páginas 58 e 59.

KLUSOWSKI, Jason M. Analyzing cart. **arXiv preprint arXiv:1906.10086**, 2019. Citado na página 24.

KORNBROT, Diana. Point biserial correlation. **Wiley StatsRef: Statistics Reference Online**, Wiley Online Library, 2014. Citado na página 75.

KWAK, Sang Kyu; KIM, Jong Hae. Statistical data preparation: management of missing values and outliers. **Korean journal of anesthesiology**, v. 70, n. 4, p. 407, 2017. Citado na página 60.

LEE, Namgil; KIM, Jong-Min. Conversion of categorical variables into numerical variables via bayesian network classifiers for binary classifications. **Computational Statistics & Data Analysis**, Elsevier, v. 54, n. 5, p. 1247–1265, 2010. Citado na página 61.

LI, Jie et al. Understanding and optimizing the gasification of biomass waste with machine learning. **Green Chemical Engineering**, Elsevier, v. 4, n. 1, p. 123–133, 2023. Citado 7 vezes nas páginas 17, 63, 64, 68, 69, 72 e 81.

LIAO, Hui et al. Machine learning prediction of supercapacitor performance of n-doped biochar from biomass wastes based on n-containing groups, element compositions, and pore structures. **Journal of Energy Storage**, Elsevier, v. 100, p. 113548, 2024. Citado na página 60.

LOBATO-PERALTA, Diego Ramón; OKOYE, Patrick U; ALEGRE, Cinthia. A review on carbon materials for electrochemical energy storage applications: State of the art, implementation, and synergy with metallic compounds for supercapacitor and battery electrodes. **Journal of Power Sources**, Elsevier, v. 617, p. 235140, 2024. Citado na página 12.

LOH, Wei-Yin. Fifty years of classification and regression trees. **International Statistical Review**, Wiley Online Library, v. 82, n. 3, p. 329–348, 2014. Citado 2 vezes nas páginas 21 e 27.

LOUPPE, Gilles. **Understanding random forests: From theory to practice**. Tese (Doutorado) — Universite de Liege (Belgium), 2014. Citado 4 vezes nas páginas 21, 25, 27 e 28.

LOZANO-GRANDE, M Azalia et al. Plant sources, extraction methods, and uses of squalene. **International journal of agronomy**, Wiley Online Library, v. 2018, n. 1, p. 1829160, 2018. Citado na página 18.

MA, Chaowei et al. Co-pyrolysis of sewage sludge and waste tobacco stem: Gas products analysis, pyrolysis kinetics, artificial neural network modeling, and synergistic effects. **Bioresource Technology**, Elsevier, v. 389, p. 129816, 2023. Citado 3 vezes nas páginas 17, 18 e 64.

MAHMOUDI-QASHQAY, Samaneh et al. Fabrication of an asymmetric supercapacitor using a novel electrode design and introduce a robust machine learning model for its performance evaluation. **Journal of Power Sources**, Elsevier, v. 613, p. 234911, 2024. Citado na página 12.

MAKSOUD, MIA Abdel et al. Advanced materials and technologies for supercapacitors used in energy conversion and storage: a review. **Environmental Chemistry Letters**, Springer, v. 19, p. 375–439, 2021. Citado na página 18.

MARCELINO, Pedro et al. Machine learning approach for pavement performance prediction. **International Journal of Pavement Engineering**, Taylor & Francis, v. 22, n. 3, p. 341–354, 2021. Citado na página 21.

MENDHE, Arpit; PANDA, Himanshu Sekhar. Machine learning-assisted electrode material fabrication and electrochemical efficiency prediction and validation of pani-ni/co hydroxide nanocomposites. **ACS Sustainable Chemistry & Engineering**, ACS Publications, v. 11, n. 49, p. 17262–17271, 2023. Citado na página 20.

MIRZA, Umar K; AHMAD, Nasir; MAJEED, Tariq. An overview of biomass energy utilization in pakistan. **Renewable and Sustainable Energy Reviews**, Elsevier, v. 12, n. 7, p. 1988–1996, 2008. Citado na página 14.

MISHRA, Sachit et al. The impact of physicochemical features of carbon electrodes on the capacitive performance of supercapacitors: a machine learning approach. **Scientific Reports**, Nature Publishing Group UK London, v. 13, n. 1, p. 6494, 2023. Citado na página 60.

MUKAKA, Mavuto M. A guide to appropriate use of correlation coefficient in medical research. **Malawi medical journal**, v. 24, n. 3, p. 69–71, 2012. Citado na página 75.

NAKATSU, Robbie T. Validation of machine learning ridge regression models using monte carlo, bootstrap, and variations in cross-validation. **Journal of Intelligent Systems**, De Gruyter, v. 32, n. 1, p. 20220224, 2023. Citado na página 40.

NANDA, Siddhartha; GHOSH, Sourav; THOMAS, Tiju. Machine learning aided cyclic stability prediction for supercapacitors. **Journal of Power Sources**, Elsevier, v. 546, p. 231975, 2022. Citado 2 vezes nas páginas 12 e 13.

NEVES, Daniel et al. Characterization and prediction of biomass pyrolysis products. **Progress in energy and combustion Science**, Elsevier, v. 37, n. 5, p. 611–630, 2011. Citado na página 16.

NG, Wartini et al. The influence of training sample size on the accuracy of deep learning models for the prediction of soil properties with near-infrared spectroscopy data. **Soil**, Copernicus Publications Göttingen, Germany, v. 6, n. 2, p. 565–578, 2020. Citado na página 60.

NGUYEN, TLU et al. Reversible addition fragmentation chain transfer (raft) polymerization in undergraduate polymer science lab. **Journal of chemical education**, ACS Publications, v. 85, n. 1, p. 97, 2008. Citado na página 18.

NOGUEIRA, Fernando. **Bayesian Optimization: Open source constrained global optimization tool for Python**. 2014–. Disponível em: https://github.com/bayesian-optimization/BayesianOptimization. Citado na página 66.

PFEIFFER, Paul E. **Probability for applications**. [S.l.]: Springer Science & Business Media, 2012. Citado na página 48.

PINTEA, Silvia L et al. A step towards understanding why classification helps regression. In: **Proceedings of the IEEE/CVF International Conference on Computer Vision**. [S.l.: s.n.], 2023. p. 19972–19981. Citado na página 66.

POTDAR, Kedar; PARDAWALA, Taher S; PAI, Chinmay D. A comparative study of categorical variable encoding techniques for neural network classifiers. **International journal of computer applications**, v. 175, n. 4, p. 7–9, 2017. Citado na página 61.

PRASAD, G Gautham et al. Supercapacitor technology and its applications: a review. In: IOP PUBLISHING. **IOP Conference Series: Materials Science and Engineering**. [S.l.], 2019. v. 561, n. 1, p. 012105. Citado na página 19.

PROBST, Philipp. **Hyperparameters, tuning and meta-learning for random forest and other machine learning algorithms**. Tese (Doutorado) — lmu, 2019. Citado 2 vezes nas páginas 42 e 58.

PROBST, Philipp; WRIGHT, Marvin N; BOULESTEIX, Anne-Laure. Hyperparameters and tuning strategies for random forest. **Wiley Interdisciplinary Reviews: data mining and knowledge discovery**, Wiley Online Library, v. 9, n. 3, p. e1301, 2019. Citado na página 58.

QI, Jun et al. On mean absolute error for deep neural network based vector-to-vector regression. **IEEE Signal Processing Letters**, IEEE, v. 27, p. 1485–1489, 2020. Citado na página 62.

QIN, Fanzhi et al. Lignocellulosic biomass carbonization for biochar production and characterization of biochar reactivity. **Renewable and Sustainable Energy Reviews**, Elsevier, v. 157, p. 112056, 2022. Citado na página 15.

QUBEISSI, Mansour Al; EL-KHAROUF, Ahmad. Renewable energy: resources, challenges and applications. IntechOpen, 2020. Citado 3 vezes nas páginas 12, 15 e 16.

RADHAKRISHNAN, Sithara et al. Borocarbonitride-based emerging materials for supercapacitor applications: Recent advances, challenges, and future perspectives. **Advanced Science**, Wiley Online Library, v. 11, n. 4, p. 2305325, 2024. Citado na página 12.

RAHIMI, Mohammad; ABBASPOUR-FARD, Mohammad Hossein; ROHANI, Abbas. Synergetic effect of n/o functional groups and microstructures of activated carbon on supercapacitor performance by machine learning. **Journal of Power Sources**, Elsevier, v. 521, p. 230968, 2022. Citado na página 60.

RAHIMI, Mohammad et al. Yield prediction and optimization of biomass-based products by multi-machine learning schemes: neural, regression and function-based techniques. **Energy**, Elsevier, v. 283, p. 128546, 2023. Citado na página 64.

RASCHKA, Sebastian. Model evaluation, model selection, and algorithm selection in machine learning. **arXiv preprint arXiv:1811.12808**, 2018. Citado na página 40.

RAVICHANDRAN, Abhilash et al. Machine learning-based prediction of cyclic voltammetry behavior of substitution of zinc and cobalt in bifeo3/bi25feo40 for supercapacitor applications. **ACS omega**, ACS Publications, v. 9, n. 31, p. 33459–33470, 2024. Citado na página 20.

RAZA, Waseem et al. Recent advancements in supercapacitor technology. **Nano Energy**, Elsevier, v. 52, p. 441–473, 2018. Citado na página 18.

ROCHA, Miguel; CORTEZ, Paulo; NEVES, José. Evolution of neural networks for classification and regression. **Neurocomputing**, Elsevier, v. 70, n. 16-18, p. 2809–2816, 2007. Citado na página 62.

SAAR-TSECHANSKY, Maytal; PROVOST, Foster. Handling missing values when applying classification models. Journal of Machine Learning Research, 2007. Citado na página 61.

SAHANI, Shalini; MAHAJAN, Hansa; HAN, Sung Soo. Unveiling the hybrid era: advancement in electrode materials for the high-performance supercapacitor: a comprehensive review. **Journal of Energy Storage**, Elsevier, v. 90, p. 111808, 2024. Citado na página 12.

ŞAHIN, Mustafa Ergin; BLAABJERG, Frede. A hybrid pv-battery/supercapacitor system and a basic active power control proposal in matlab/simulink. **Electronics**, MDPI, v. 9, n. 1, p. 129, 2020. Citado 3 vezes nas páginas 7, 18 e 19.

ŞAHIN, Mustafa Ergin; BLAABJERG, Frede; SANGWONGWANICH, Ariya. A comprehensive review on supercapacitor applications and developments. **Energies**, MDPI, v. 15, n. 3, p. 674, 2022. Citado 3 vezes nas páginas 12, 18 e 19.

SARAMOLEE, Prachid et al. From a sustainable natural rubber sponge to an activation-free sulfur-rich biocarbon sponge as a potential electrode material for a supercapacitor. **Chemical Engineering Journal**, Elsevier, v. 503, p. 158231, 2025. Citado na página 20.

SAWANT, Digambar S et al. Transition metal molybdates emerging materials for high-performance supercapacitors: A machine learning analysis. **Battery Energy**, Wiley Online Library, p. e20240073, 2025. Citado na página 20.

SAWANT, Vaishali; DESHMUKH, Rashmi; AWATI, Chetan. Machine learning techniques for prediction of capacitance and remaining useful life of supercapacitors: A comprehensive review. **Journal of Energy Chemistry**, Elsevier, v. 77, p. 438–451, 2023. Citado na página 13.

SCHOBER, Patrick; BOER, Christa; SCHWARTE, Lothar A. Correlation coefficients: appropriate use and interpretation. **Anesthesia & analgesia**, LWW, v. 126, n. 5, p. 1763–1768, 2018. Citado na página 65.

SCORNET, Erwan. On the asymptotics of random forests. **Journal of Multivariate Analysis**, Elsevier, v. 146, p. 72–83, 2016. Citado 2 vezes nas páginas 44 e 49.

SHARIFANI, Koosha; AMINI, Mahyar. Machine learning and deep learning: A review of methods and applications. **World Information Technology and Engineering Journal**, v. 10, n. 07, p. 3897–3904, 2023. Citado na página 21.

SINGH, Anita; AGRAWAL, Madhoolika. Acid rain and its ecological consequences. **Journal of environmental biology**, PRAGATI PRESS-INDIA, v. 29, n. 1, p. 15, 2007. Citado na página 17.

SLAMERŠAK, Aljoša; KALLIS, Giorgos; O'NEILL, Daniel W. Energy requirements and carbon emissions for a low-carbon energy transition. **Nature communications**, Nature Publishing Group UK London, v. 13, n. 1, p. 6932, 2022. Citado na página 14.

SNOEK, Jasper; LAROCHELLE, Hugo; ADAMS, Ryan P. Practical bayesian optimization of machine learning algorithms. **Advances in neural information processing systems**, v. 25, 2012. Citado 2 vezes nas páginas 58 e 59.

SUTTON, Clifton D. Classification and regression trees, bagging, and boosting. **Handbook of statistics**, Elsevier, v. 24, p. 303–329, 2005. Citado 8 vezes nas páginas 21, 22, 23, 24, 29, 39, 40 e 41.

SZEGHALMY, Szilvia; FAZEKAS, Attila. A comparative study of the use of stratified cross-validation and distribution-balanced stratified cross-validation in imbalanced learning. **Sensors**, MDPI, v. 23, n. 4, p. 2333, 2023. Citado na página 66.

UDDIN, Md N et al. An overview of recent developments in biomass pyrolysis technologies. **Energies**, MDPI, v. 11, n. 11, p. 3115, 2018. Citado 3 vezes nas páginas 14, 15 e 16.

VABALAS, Andrius et al. Machine learning algorithm validation with a limited sample size. **PloS one**, Public Library of Science San Francisco, CA USA, v. 14, n. 11, p. e0224365, 2019. Citado na página 59.

VIERING, Tom; LOOG, Marco. The shape of learning curves: a review. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, IEEE, v. 45, n. 6, p. 7799–7819, 2022. Citado na página 59.

WANG, Liang; CARVALHO, Luis. Multiclass roc. **arXiv preprint arXiv:2404.13147**, 2024. Citado na página 62.

WRIGHT, Stephen; NOCEDAL, Jorge et al. Numerical optimization. **Springer Science**, v. 35, n. 67-68, p. 7, 1999. Citado na página 52.

WU, Jia et al. Hyperparameter optimization for machine learning models based on bayesian optimization. **Journal of Electronic Science and Technology**, Elsevier, v. 17, n. 1, p. 26–40, 2019. Citado na página 58.

YAKUBU, Rahimat Oyiza et al. Performance analysis of a 50 mw solar pv installation at bui power authority: A comparative study between sunny and overcast days. **Electricity**, MDPI, v. 5, n. 3, p. 546–561, 2024. Citado na página 12.

YANG, Hao et al. Machine learning guided 3d printing of carbon microlattices with customized performance for supercapacitive energy storage. **Carbon**, Elsevier, v. 201, p. 408–414, 2023. Citado na página 20.

YATES, Luke A et al. Cross validation for model selection: a review with examples from ecology. **Ecological Monographs**, Wiley Online Library, v. 93, n. 1, p. e1557, 2023. Citado 2 vezes nas páginas 41 e 42.

YESILBAG, Yasar Ozkan et al. The hierarchical synthesis of tungsten disulfide coated vertically aligned boron carbon nitride nanotubes composite electrodes for supercapacitors. **Journal of Energy Storage**, Elsevier, v. 52, p. 104964, 2022. Citado na página 79.

YU, Jie; ODRIOZOLA, José A; REINA, Tomas R. Dry reforming of ethanol and glycerol: mini-review. **Catalysts**, MDPI, v. 9, n. 12, p. 1015, 2019. Citado na página 16.

ZHANG, Wengang et al. Prediction of undrained shear strength using extreme gradient boosting and random forest based on bayesian optimization. **Geoscience Frontiers**, Elsevier, v. 12, n. 1, p. 469–477, 2021. Citado na página 58.

ZHAO, Jingyuan; BURKE, Andrew F. Review on supercapacitors: Technologies and performance evaluation. **Journal of energy chemistry**, Elsevier, v. 59, p. 276–291, 2021. Citado 2 vezes nas páginas 18 e 19.

UNIVERSIDADE DO ESTADO DE SANTA CATARINA – UDESC
BIBLIOTECA UNIVERSITÁRIA
REPOSITÓRIO INSTITUCIONAL

CENTRO DE CIÊNCIAS TECNOLÓGICAS – CCT

ATESTADO DE VERSÃO FINAL

Eu, Ademir Nied, professor do curso de Mestrado Acadêmico em Engenharia Elétrica, declaro que esta é a versão final aprovada pela comissão julgadora da dissertação intitulada: **“**APLICAÇÃO DE ALGORITMOS DE APRENDIZAGEM DE MÁQUINA OTIMIZADOS NA PREVISÃO DE PROCESSOS TERMOQUÍMICOS E DE CAPACITÂNCIA DE SUPERCAPACITORES**”** de autoria do acadêmico Lucas Lima Nogueira.

Joinville, 13 de fevereiro de 2026.

Assinatura digital do(a) orientador(a):

Documento assinado digitalmente
gov.br
ADEMIR NIED
Data: 26/02/2026 15:28:04-0300
Verifique em https://validar.iti.gov.br

_____

Ademir Nied