CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA “PAULA SOUZA”

FACULDADE DE TECNOLOGIA DE BEBEDOURO

TECNOLOGIA EM BIG DATA NO AGRONEGÓCIO

PREVISÃO DE PRODUÇÃO DE CANA-DE-AÇÚCAR UTILIZANDO FRAMEWORK LOW-CODE PYCARET

AUTOR: PAULO SERGIO CALOR JUNIOR

ORIENTADOR: PROF. DR. RENAN GUILHERME NESPOLO

BEBEDOURO

2025

PAULO SERGIO CALOR JUNIOR

PREVISÃO DE PRODUÇÃO DE CANA-DE-AÇÚCAR UTILIZANDO FRAMEWORK LOW-CODE PYCARET

Monografia apresentada à Faculdade de Tecnologia de Bebedouro, como parte dos requisitos para a obtenção do título de Tecnólogo em Big Data no Agronegócio

Orientador: **prof. Dr. Renan Guilherme Nespolo**

BEBEDOURO

2025

“Não se gerencia o que não se mede, não se mede o que não se define, não se define o que não se entende, e não há sucesso no que não se gerencia.”

William Edwards Deming

AGRADECIMENTOS

Agradeço, primeiramente, a Deus, por me conduzir e iluminar nesta jornada de grande aprendizado e conhecimento. Sua força foi essencial para atravessar os desertos e superar os desafios que se apresentaram no caminho.

Com muito carinho e gratidão, dedico esta conquista à minha amada família. Em especial, à minha esposa, Licia, e aos meus filhos, Benicio e Bella. Vocês sempre acreditaram em mim e me apoiaram nesta caminhada. A paciência, o amor e o incentivo de vocês foram a base fundamental para que eu chegasse até aqui.

Ao meu orientador, Prof. Dr. Renan, expresso meu sincero reconhecimento por todo o seu apoio, paciência e valiosos ensinamentos ao longo destes anos. Seu conhecimento técnico e sua dedicação foram indispensáveis para a realização e a qualidade deste trabalho.

Aos meus amigos, Guilherme e Júlio, minha eterna gratidão por estarem sempre ao meu lado em cada etapa deste ciclo. O apoio, a ajuda e, muitas vezes, o encorajamento nos momentos mais necessários foram de um valor inestimável.

Estendo meus agradecimentos aos meus amigos de trabalho, Bruno e Kairo, que me acolheram e me concederam uma oportunidade fundamental nesta nova fase da minha vida, agregando conhecimento e me ensinando a enfrentar os desafios da profissão.

A todos que, direta ou indiretamente, contribuíram para a conclusão deste projeto, meu muito obrigado.

CALOR JUNIOR, P. S. **Previsão de Produção de Cana-de-Açúcar Utilizando Framework Low-Code Pycaret**. Trabalho de Graduação (Monografia). Centro Estadual de Educação Tecnológica “Paula Souza”. Faculdade de Tecnologia de Bebedouro. nº p. 2025.

RESUMO

No presente trabalho é a apresentado um estudo sobre o *framework low-code* PyCaret. Esse *framework* seleciona modelos de Aprendizado de Máquina de forma automatizada (AutoML, englobando pré-processamento, seleção de características e previsão de resultados. No campo do Agronegócio uma das possíveis aplicações do *framework* é a previsão de produtividade agrícola, medida em toneladas de cana por hectare (TCH), em usinas brasileiras. Deste modo o presente estudo concentrou-se em aplicar o framework *low-code* para realizar a previsão do TCH, que realizou a avaliação de vários modelos de regressão para esse problema, considerando variáveis agronômicas, climáticas e operacionais. O estudo ainda apresentou uma comparação entre os dois métodos que apresentaram os melhores desempenho, segundo o *framework*, a partir das métricas de avaliação adotadas. Os resultados apresentados após a validação do modelo, em cenários agrícolas reais, indicaram a presença de *outliers* e outros problemas que explicaram a seleção de determinados algoritmos. O estudo contribui para a compreensão do uso de AutoML na agricultura de precisão apresentando algoritmos, validação de modelos e limitações do uso do *framework*.

Palavras-chave: Aprendizado de Máquina Automatizado; TCH; Cana-de-açúcar; PyCaret; *Huber Regressor*; Agricultura de Precisão.

CALOR JUNIOR, P. S. **Previsão de Produção de Cana-de-Açúcar Utilizando Framework Low-Code Pycaret**. Trabalho de Graduação (Monografia). Centro Estadual de Educação Tecnológica “Paula Souza”. Faculdade de Tecnologia de Bebedouro. nº p. 2025.

ABSTRACT

This paper presents a study on the low-code PyCaret framework. This framework selects Machine Learning models in an automated way (AutoML, encompassing preprocessing, feature selection, and result prediction. In the Agribusiness field, one of the possible applications of the framework is the prediction of agricultural productivity, measured in tons of sugarcane per hectare (TCH), in Brazilian mills. Thus, the present study focused on applying the low-code framework to perform the TCH prediction, which performed an evaluation of several regression models for this problem, considering agronomic variables. The study also presented a comparison between the two methods that presented the best performances, according to the framework, based on the mandatory evaluation analyzes.

Keywords: Automated Machine Learning; TCH; Sugarcane; PyCaret; Huber Regressor; Precision Agriculture.

SUMÁRIO

LISTA DE FIGURAS

Lista de Tabelas

Lista de Gráficos

Lista de Algoritmos

LISTA DE EQUAÇÕES

Lista de Siglas

TCH: Toneladas de Cana por Hectare

-ML: *Machine learning* (Aprendizado de Máquina)

-MAE: Mean Absolute Error (Erro Médio Absoluto)

- RMSE: Root Mean Square Error (Raiz do Erro Quadrático Médio)

-R^2: Coeficiente de Determinação

-MAPE: Mean Absolute Percentage Error (Erro Percentual Absoluto Médio)

-CV: Cross-Validation (Validação Cruzada)

-API: Application Programming Interface

-CSV: Comma-Separated Values

-HTM: HyperText Markup Language

JSON: JavaScript Object Notation

SQL: Structured Query Language

ATR: Açúcar Total Recuperável

TCH: Toneladas de Cana por Hectare

-GPS: Global Positioning System

-IoT: Internet of Things (Internet das Coisas)

1 INTRODUÇÃO

A cana-de-açúcar *Saccharum officinarum L.* representa uma das culturas agrícolas mais importantes do Brasil, ocupando posição de destaque no cenário mundial de produção de açúcar e bioenergia. Segundo dados da Companhia Nacional de Abastecimento (CONAB), a safra 2024/25 de cana-de-açúcar no Brasil apresentou produção estimada em 676,96 milhões de toneladas, consolidando o país como maior produtor mundial desta cultura. O setor sucroenergético brasileiro é responsável por aproximadamente 2% do PIB nacional e gera mais de 800 mil empregos diretos, demonstrando sua relevância econômica e social (CONAB, 2024).

Neste contexto, a predição precisa da produtividade, medida por meio da métrica Toneladas de Cana por Hectare (THC), constitui um desafio científico e tecnológico fundamental para otimização dos processos produtivos. O TCH representa o principal indicador de produtividade utilizado no setor sucroenergético, influenciando diretamente decisões estratégicas relacionadas ao planejamento da safra, dimensionamento da capacidade industrial, logística de transporte e viabilidade econômica dos empreendimentos (SCARPARI, 2002; PEDROZO, 2017; ATVOS, 2022).

A complexidade inerente aos sistemas agrícolas, caracterizada pela interação de múltiplos fatores bióticos e abióticos, torna a predição de produtividade um problema multivariado de alta dimensionalidade. Variáveis como características edafoclimáticas, práticas de manejo, variedades genéticas, histórico de cultivo e condições meteorológicas interagem de forma não-linear, criando padrões complexos que desafiam métodos tradicionais de análise estatística.

1.1 JUSTIFICATIVA

O problema reside na necessidade de desenvolver métodos computacionais avançados capazes de capturar e modelar as relações entre variáveis para predição do TCH. Métodos tradicionais de estimativa, baseados em médias históricas ou regressões lineares simples, frequentemente apresentam limitações significativas quando aplicados a dados agrícolas reais, caracterizados por alta variabilidade e relações não-lineares (GREGO, 2020).

A emergência de técnicas de Aprendizado de Máquina (*Machine learning*) oferece oportunidades sem precedentes para abordar esta problemática, permitindo a descoberta automática de padrões complexos em grandes volumes de dados. Entretanto, a aplicação efetiva destas técnicas requer expertise técnica especializada, criando uma barreira significativa para

adoção no setor agrícola (WHITE *et* al., 2021).

O Aprendizado de Máquina Automatizado (AutoML) surge como uma solução promissora para analisar o acesso a estas tecnologias avançadas, automatizando tarefas complexas como seleção de algoritmos, otimização de hiper parâmetros e validação de modelos. O *framework* PyCaret, desenvolvida especificamente para facilitar a implementação de workflows de aprendizado de máquina, representa uma ferramenta particularmente adequada para esta finalidade (ALI, 2020; PAREKH, 2025).

1.2 OBJETIVOS

Desenvolver e validar cientificamente um sistema de predição de TCH baseado em técnicas de Aprendizado de Máquina Automatizado, utilizando o *framework* PyCaret para comparação sistemática de algoritmos de regressão e implementação de interface web acessível para aplicação prática no setor sucroenergético. Deste modo os objetivos do presente estudo são:

1. Analisar tecnicamente a arquitetura do *framework* PyCaret, identificando os mecanismos internos de seleção automática de algoritmos de aprendizado de máquina.
2. Comparar sistematicamente os 2 melhores algoritmos de regressão de acordo com as métricas estabelecidas no *framework*.
3. Realizar validação dos modelos dos algoritmos utilizando métricas adotadas em problemas de regressão.

1.3 ESTRUTURA DO TRABALHO

Esta monografia está estruturada em cinco capítulos principais, organizados de forma a apresentar uma progressão lógica desde a fundamentação teórica até a aplicação prática dos resultados. O Capítulo 2 apresenta uma fundamentação teórica abrangente sobre técnicas de regressão, AutoML e a *framework* PyCaret, estabelecendo as bases científicas necessárias para compreensão do trabalho.

O Capítulo 3 detalha a metodologia utilizada, incluindo descrição do *dataset*, configuração dos experimentos e critérios de avaliação. O Capítulo 4 apresenta os resultados obtidos e discussão científica dos achados, incluindo análise quantitativa das métricas de performance e interpretação da importância das variáveis.

O Capítulo 5 sintetiza as principais conclusões do trabalho e propõe direções para pesquisas futuras. Elementos pós-textuais incluem referências bibliográficas, glossário de termos técnicos e anexos com código fonte comentado, proporcionando transparência e reprodutibilidade dos resultados.

2 FUNDAMENTAÇÃO TEÓRICA

O presente Capítulo apresenta toda a fundamentação deste estudo, segmentados em: 2.1 Métodos de Aprendizado em Conjunto (Ensemble Learning); 2.2 Aprendizado de Máquina Automatizado (AutoML); 2.3 Framework PyCaret: Arquitetura e Funcionalidades; 2.4 Técnicas de Pré-processamento de Dados; 2.5 Algoritmos de Regressão Aplicados; e 2.6 Metodologia de Avaliação de Modelos.

2.1 MÉTODOS DE APRENDIZADO EM CONJUNTO (*ENSEMBLE LEARNING*)

No campo do aprendizado de máquina, raramente um único modelo consegue capturar toda a complexidade contida em um conjunto de dados. A performance de um modelo individual é limitada pelo seu próprio viés (erros sistemáticos devido a suposições simplificadoras) e variância (sensibilidade a pequenas flutuações nos dados de treinamento). Os métodos de aprendizado em conjunto, ou *ensemble learning*, surgem como uma poderosa meta-abordagem para mitigar essas limitações. A premissa central é que, ao combinar estrategicamente as predições de múltiplos modelos (chamados de "aprendizes fracos" ou *weak learners*), é possível obter um "super modelo" com uma capacidade de generalização e uma acurácia preditiva superiores às de qualquer um dos seus componentes individuais (HASTIE; TIBSHIRANI; FRIEDMAN, 2009).

Existem diversas estratégias para construir um *ensemble*, sendo as mais proeminentes o *Bagging*, o *Boosting* e o *Stacking.*

2.1.1 BAGGING (BOOTSTRAP AGGREGATING)

O *Bagging*, abreviação de *Bootstrap Aggregating*, é uma técnica de *ensemble* que visa principalmente à redução da variância de um modelo. O processo, proposto por Breiman (1996), consiste em gerar B subconjuntos de dados a partir do conjunto de treinamento original, por meio de amostragem com reposição (*bootstrap*). Para cada subconjunto, um modelo aprendiz $f\_b(x)$ é treinado de forma independente e paralela. Em tarefas de regressão, como a predição de TCH, a predição final do *ensemble*, $\hat{f}_{bag}(x)$, é tipicamente a média das predições de todos os modelos individuais, conforme a (Equação 1).

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} f_b(x) \quad (1)$$

A Equação 1 demonstra que, ao agregar os resultados de B modelos treinados em diferentes amostras dos dados, o *Bagging* suaviza as predições e torna o modelo final mais estável e menos propenso a sobreajuste (*overfitting*). O algoritmo *Random Forest* é um exemplo clássico e poderoso de aplicação do método *Bagging* (Breiman; 1996).

2.1.2 BOOSTING

Diferentemente do *Bagging*, o *Boosting* é uma técnica de *ensemble* que treina modelos de forma sequencial e adaptativa, com o objetivo principal de reduzir o viés do modelo preditivo.

O processo, formalizado por Friedman (2001), inicia-se com um modelo simples e, a cada iteração m, adiciona um novo aprendiz $h_m(x)$, treinado para corrigir os erros residuais do modelo anterior. Assim, a predição do *ensemble* na etapa m é uma versão aprimorada da predição na etapa $m-1$, como descrito na forma aditiva (Equação 2):

$$\mathrm{F_m(x)} = \mathrm{F}_{m-1}(\mathrm{x}) + \nu \mathrm{h_m(x)} \quad (2)$$

Nessa equação, $\mathrm{F_m(x)}$ representa o modelo do *ensemble* na iteração m e ν (taxa de aprendizado, *learning rate*) é um parâmetro que controla a contribuição de cada novo aprendiz ao modelo final. Ao construir uma cadeia de modelos que aprendem com os erros dos anteriores, o *Boosting* consegue criar um *ensemble* de alta performance, capaz de alcançar elevada acurácia preditiva mesmo em problemas complexos.

Algoritmos como o *Light Gradient Boosting Machine (LightGBM)*, um dos modelos centrais deste trabalho, são exemplos proeminentes dessa família. O *LightGBM* utiliza árvores de decisão como aprendizes fracos e implementa técnicas inovadoras para otimizar a eficiência e a escalabilidade, como o *Gradient-based One-Side Sampling (GOSS)* e o *Exclusive Feature Bundling (EFB)*. Essas estratégias permitem ao *LightGBM* lidar com grandes volumes de dados, mantendo alta velocidade de treinamento e precisão (Friedman; 2001).

2.1.3 *STACKING* (GENERALIZAÇÃO EMPILHADA)

O *Stacking*, ou generalização empilhada, é uma técnica de *ensemble* que leva a combinação de modelos a outro nível de abstração. O processo é organizado em camadas. Na primeira camada (nível 0), múltiplos modelos de diferentes tipos como um modelo linear, uma árvore de decisão ou uma máquina de vetores de suporte são treinados de forma independente. As predições geradas por esses modelos de base são então utilizadas como variáveis de entrada para treinar um segundo modelo, denominado "meta-aprendiz" (nível 1), cuja função é aprender a melhor forma de combinar as predições dos modelos de base para gerar a predição final (WOLPERT, 1992).

Diferentemente do *Bagging* e do *Boosting*, o *Stacking* normalmente utiliza modelos de base heterogêneos, ou seja, de diferentes tipos, aproveitando as diferentes perspectivas que cada modelo pode oferecer sobre os dados. O meta-aprendiz pode ser qualquer algoritmo de aprendizado supervisionado, como regressão linear, árvore de decisão ou até mesmo redes neurais, dependendo da complexidade do problema.

A arquitetura do *Stacking* permite que o modelo final supere a performance de qualquer modelo individual do conjunto, justamente por aprender a explorar as complementaridades entre eles (WOLPERT, 1992).

2.2 APRENDIZADO DE MÁQUINA AUTOMATIZADO (AUTOML)

A existência de uma vasta gama de algoritmos, incluindo os complexos métodos de *ensemble* discutidos anteriormente, torna a tarefa de selecionar, configurar e otimizar manualmente o melhor modelo para um dado problema um processo árduo, demorado e que exige profundo conhecimento especializado. É para este desafio que emerge o paradigma do Aprendizado de Máquina Automatizado (AutoML).

O AutoML representa uma mudança de paradigma na prática da ciência de dados, buscando automatizar o processo de ponta a ponta do desenvolvimento de modelos de *machine learning*. Esta abordagem surge como resposta à crescente complexidade dos *workflows* e à necessidade de democratizar o acesso a estas tecnologias. O escopo do AutoML, conforme definido pela literatura, abrange a automação de todo o *pipeline*, incluindo o pré-processamento, a seleção do algoritmo, a otimização de seus hiper parâmetros e a construção de *ensemble*s (HE; ZHAO; CHU, 2021). O objetivo fundamental é transformar o processo, tradicionalmente artesanal, em uma busca sistemática, eficiente e mais acessível.

2.2.1 ARQUITETURAS E ABORDAGENS DE AUTOML

Diferentes estratégias foram propostas para realizar a busca no vasto espaço de possíveis *pipelines* de *machine learning*. As principais abordagens podem ser categorizadas da seguinte forma:

Otimização *Bayesiana*: Esta é uma das abordagens mais sofisticadas e eficientes, especialmente para a otimização de hiper parâmetros. Em vez de uma busca cega, ela utiliza um modelo probabilístico (um "modelo substituto", tipicamente um Processo Gaussiano) para mapear a relação entre as configurações de hiper parâmetros e a performance do modelo. A cada iteração, uma função de aquisição é usada para decidir qual a próxima configuração de hiper parâmetros a ser avaliada, balanceando de forma inteligente a exploração de novas áreas do espaço de busca (*exploration*) e a otimização em torno das melhores soluções já encontradas (*exploitation*) (SNOEK; LAROCHELLE; ADAMS; 2012). Uma função de aquisição comum é a Melhora Esperada (*Expected Improvement* - EI), definida na Equação 3.

$$\mathrm{EI}(\mathrm{x}) = \mathrm{E}[\max(\mathrm{f}(\mathrm{x}) - \mathrm{f}(\mathrm{x}^{+}), 0)] \tag{3}$$

no qual $f(x)$ é o valor da função objetivo (ex: RMSE) no ponto x e $f(x^{+})$ é o melhor valor observado até o momento. A função busca maximizar a melhora esperada em relação ao melhor ponto atual.

Algoritmos Evolutivos: Inspirados na teoria da evolução biológica, estes métodos mantêm uma "população" de *pipelines* de *machine learning* candidatos. A cada "geração", os *pipelines* de melhor desempenho são selecionados (seleção) e combinados entre si para gerar novos e promissores *pipelines* (cruzamento ou *crossover*), enquanto pequenas alterações aleatórias são introduzidas para manter a diversidade (mutação). Esse processo evolutivo permite explorar o espaço de busca de forma paralela e robusta.

Meta-Aprendizado (*Meta-Learning*): Esta abordagem visa "aprender a aprender". Ela utiliza conhecimento adquirido em experimentos de *machine learning* realizados em problemas e *datasets* anteriores para acelerar a otimização em um novo problema. O sistema analisa as características do novo *dataset* (suas "*meta-features*", como número de amostras, de variáveis, etc.) e recomenda uma configuração inicial de algoritmo e hiper parâmetros que funcionou bem em *datasets* semelhantes no passado, tornando a busca muito mais eficiente.

2.2.2 DESAFIOS E LIMITAÇÕES DO AUTOML

Apesar de sua potência, o AutoML não é uma solução universal. Primeiramente, a interpretação dos modelos gerados pode ser um obstáculo, pois sistemas AutoML frequentemente convergem para *ensembles* complexos que funcionam como "caixas-pretas". Em segundo lugar, o custo computacional pode ser um limitador. Finalmente, o AutoML é sensível à qualidade dos dados que não pode compensar problemas como viés de amostragem ou erros de medição, sendo a curadoria dos dados uma etapa que ainda depende fundamentalmente da expertise humana, especialmente em domínios com dados complexos ou incompletos (LIM *et* al., 2024).

Ferramentas modernas, como a utilizada neste trabalho, buscam mitigar esses desafios. Elas implementam uma combinação híbrida dessas abordagens, utilizando, por exemplo, o meta-aprendizado para sugerir um ponto de partida e a otimização bayesiana para o ajuste fino, tornando o processo mais eficiente. Diante dessas estratégias e desafios, diversas ferramentas foram desenvolvidas para aplicar os princípios do AutoML na prática. Neste trabalho, a biblioteca escolhida para esta finalidade foi o *Framework* PyCaret, cujas características e arquitetura são detalhadas a seguir(LIM *et* al., 2024)..

2.3 FRAMEWORK PYCARET: ARQUITETURA E FUNCIONALIDADES

Para operacionalizar os conceitos de Aprendizado de Máquina Automatizado (AutoML) em um fluxo prático, sistemático e reprodutível para a predição de TCH, este estudo adotou a biblioteca *open-source* PyCaret. Seu *design* centrado na produtividade e abordagem *low-code* foram decisivos para a condução dos experimentos. O PyCaret foi desenvolvido com a filosofia de democratizar o acesso a técnicas avançadas de *machine learning*, permitindo que o pesquisador foque na análise do problema e na interpretação dos resultados, em vez de lidar com a complexidade da codificação (PAREKH, 2023).

A robustez da biblioteca deriva de sua arquitetura, construída sobre pilares consolidados do ecossistema *Python* de ciência de dados, como *scikit-learn*, *pandas*, *numpy*, *matplotlib* e *plotly*, apresentadas na Tabela 1. O PyCaret atua como um orquestrador de alto nível, integrando essas bibliotecas para garantir confiabilidade, escalabilidade e desempenho ao fluxo preditivo (PAREKH, 2023).

Tabela 1 - Bibliotecas *Python* necessárias para o funcionamento do *framework* PyCaret.

Bibliotecas necessárias
Scikit-learn
Pandas
Numpy
Matplotlib
Plotly

Fonte: (referência).

2.3.1 CAMADA DE ABSTRAÇÃO E *PIPELINE* DE PRÉ-PROCESSAMENTO

A principal característica arquitetônica do PyCaret é sua camada de abstração, que unifica as interfaces de diferentes bibliotecas, permitindo ao usuário treinar, avaliar e utilizar modelos de múltiplas origens (por exemplo, *scikit-learn, LightGBM, CatBoost, XGBoost*) por meio de um conjunto de comandos consistente. Essa abordagem reduz significativamente a complexidade e a curva de aprendizado associadas ao uso de cada biblioteca individual (ALI, 2020).

Essa abstração é operacionalizada pela função *setup()*, ponto central de qualquer experimento no PyCaret. Essa função inicializa o ambiente experimental e constrói um *pipeline* de pré-processamento automatizado e customizável. Ao ser executada, ela infere os tipos de dados, trata valores ausentes, codifica variáveis categóricas, normaliza as *features* e realiza outras tarefas preparatórias essenciais. Cada uma dessas etapas é encapsulada em um objeto *pipeline* do *scikit-learn* (PEDREGOSA *et* al., 2011), garantindo que qualquer novo dado apresentado ao modelo para predição passe exatamente pela mesma sequência de transformações aplicadas aos dados de treinamento. Essa abordagem é fundamental para evitar vazamento de dados (*data leakage*) e assegurar a integridade e reprodutibilidade do modelo.

A Tabela 2 detalha os principais parâmetros da função *setup()* utilizados neste estudo.

Tabela 2 - Principais parâmetros da função *setup()* do PyCaret

Parâmetro	Obrigatoriedade	Descrição
data	Obrigatório	*DataFrame* do *pandas* contendo o conjunto de dados completo a ser utilizado no experimento.
target	Obrigatório	*String* com o nome da coluna que se deseja prever (a variável alvo).
session_id	Opcional	Semente aleatória (*seed*) para garantir a reprodutibilidade dos resultados. Experimentos com o mesmo *session_id* e dados produzirão saídas idênticas.
train_size	Opcional	Proporção do *dataset* (ex: 0.3 para 70%) a ser usada para treinamento e validação cruzada. O restante é separado como um conjunto de teste (*Hold-out*).
normalize	Opcional	Se *True*, aplica uma normalização (como Z-score) às variáveis numéricas para que tenham uma escala comum, melhorando o desempenho de certos algoritmos.
transformation	Opcional	Se *True*, aplica uma transformação de potência (como Yeo-Johnson) para tornar a distribuição das variáveis mais próxima de uma gaussiana (normal).
remove_multicollinearity	Opcional	Se *True*, remove variáveis preditoras que são altamente correlacionadas entre si, o que pode evitar problemas de instabilidade no modelo.
multicollinearity_threshold	Opcional	Limiar de correlação de Pearson (ex: 0.9) acima do qual uma variável é considerada multicolinear e removida.
fold	Opcional	Número de partições (*folds*) a serem usadas na validação cruzada, que é a técnica utilizada para avaliar o desempenho do modelo de forma robusta.

Fonte: (referência).

O PyCaret também integra funcionalidades de rastreamento de experimentos, como integração com *MLflow*, permitindo o registro automático de métricas, hiper parâmetros e artefatos de modelos, o que facilita a comparação entre diferentes abordagens e reforça a reprodutibilidade dos resultados. O *framework* é estruturado em módulos especializados para classificação, regressão, *clustering*, detecção de anomalias, processamento de linguagem natural e séries temporais, cada um com funções e *pipelines* próprios, otimizando o *workflow* para diferentes tipos de problemas (PAREKH, 2025; PYCARET, 2022).

Em suma, a utilização do PyCaret possibilitou a criação de um *workflow* de *machine learning* rigoroso, automatizado e reprodutível. Após a configuração do ambiente experimental via função *setup()*, foi possível seguir com a definição da metodologia de avaliação, etapa detalhada na próxima seção.

2.4 PRÉ-PROCESSAMENTO

No presente estudo o pré-processamento utilizado pelo framework apresentado na Seção 2.3 a transformação de Yeo-Johnson, utilizada para normalizar distribuições não-gaussianas,

$$\psi(\lambda, y) = \begin{cases} \dfrac{(y+1)\lambda - 1}{\lambda}, 1\ se\ \lambda \neq 0\ e\ y \geq 0 \\ log(y+1), se\ \lambda = 0\ e\ y \geq 0 \\ -\dfrac{(-y+1)^{2-\lambda} - 1}{2-\lambda}, se\ \lambda \neq 2\ e\ y < 0 \\ -\log(-y+1), se\ \lambda = 2\ e\ y < 0 \end{cases} \quad (4)$$

No qual $\lambda(lambda)$: é o parâmetro de transformação que é ajustado (otimizado) para encontrar a melhor normalização para a distribuição dos dados; y representa o valor original da observação no conjunto de dados

A Equação 4, transformação de Yeo-Johnson, uma técnica estatística utilizada para normalizar variáveis que podem conter valores negativos, zero ou positivos YEO; JOHNSON (2000), dado que: se $\lambda \neq 0$, aplica uma transformação de potência ajustada pelo parâmetro λ; se $\lambda = 0$, transformação logarítmica para evitar problemas com $y = 0$; se $\lambda \neq 2$, transformação de potência invertida, garantindo continuidade e suavidade e; se $\lambda = 2$, transformação logarítmica invertida.

Esse pré-processamento é adotado para tratar valores ausentes e remoção de *outliers* extremos, definidos pelo critério do Intervalo Interquartil (IQR) TUKEY (1977), apresentado na Equação 5:

$$outlier > Q3 + 1{,}5 \times IQR\ ou\ outlier < Q1 - 1{,}5 \times IQR \quad (5)$$

No qual $Q3$ é o terceiro quartil, $Q1$ é o primeiro quartil e IQR é o intervalo Interquartil.

2.5 HUBER REGRESSOR

O *Huber Regressor* é um modelo de regressão linear robusto, especialmente projetado para lidar com a presença de *outliers* nos dados. Diferentemente da regressão por Mínimos Quadrados Ordinários (OLS), o modelo de Huber emprega uma função de perda híbrida que combina as melhores características do Erro Quadrático Médio (MSE) e do Erro Absoluto Médio (MAE): para erros de pequena magnitude, a penalização é quadrática; para erros grandes, a penalização se torna linear (HUBER, 1964).

A seguir, apresenta-se a expressão matemática da função de perda de Huber, conforme Equação 6

$$L = \begin{cases} \frac{1}{2}\left\|y - \bar{x}^T\bar{\theta}\right\|_2^2 \; se \; |y - \bar{x}^T\bar{\theta}| \leq t_H \\ t_H|y - \bar{x}^T\bar{\theta}\left| - \frac{t_H}{2} \right. \quad caso \; contrário \end{cases}, \tag{6}$$

no qual $\bar{x}$ é o vetor de entrada (features); $\bar{\theta}$ é o vetor de parâmetros do modelo θ; $\bar{x}^T\bar{\theta}$ é dado como produto interno; y é o valor real; t_H dado como limite que define a transição entre erro quadrático e linear.

Essa transição entre o comportamento quadrático e linear é controlada por um hiper parâmetro, geralmente denotado como épsilon (ϵ). Esse parâmetro define o limiar do erro: resíduos menores que ϵ são penalizados de forma quadrada, enquanto resíduos maiores são penalizados linearmente. A otimização desse hiper parâmetro, realizada pelo PyCaret, permite ao modelo aprender a partir dos dados qual é o nível de desvio que deve ser considerado um *outlier*.

A performance superior do *Huber Regressor* neste estudo é um forte indicativo de que o conjunto de dados, embora tratado, ainda contém uma variabilidade que se afasta da distribuição normal. Sugere que os *outliers* presentes não são meramente erros de medição, mas sim representações de eventos agronômicos reais e extremos (um ataque localizado de pragas, uma falha pontual de irrigação ou uma mancha de solo com fertilidade anômala), cuja influência sobre o modelo precisa ser moderada, mas não ignorada.

2.6 LIGHT GRADIENT BOOSTING MACHINE (LIGHTGBM)

O *LightGBM* é um algoritmo de *ensemble* da família dos métodos de *Gradient Boosting*, que constrói modelos baseados em árvores de decisão de forma sequencial. Em cada iteração, uma nova árvore é treinada para corrigir os erros residuais do conjunto de árvores anteriores, otimizando assim a função de perda global.

Desenvolvido por Ke *et* al. (2017), o *LightGBM* se destaca por sua eficiência computacional e alta performance preditiva, sendo amplamente adotado em competições e aplicações industriais. Suas principais inovações metodológicas incluem: GOSS (*Gradient-based One-Side Sampling*): uma estratégia inteligente de amostragem baseada na magnitude dos gradientes dos exemplos de treino; EFB (*Exclusive Feature Bundling*): técnica que agrupa variáveis esparsas e mutuamente exclusivas, reduzindo a dimensionalidade do problema e acelerando o treinamento; Crescimento "*leaf-wise*": ao contrário do crescimento tradicional "nivelado" (*level-wise*), o *LightGBM* expande preferencialmente as folhas que mais reduzem a perda, resultando em convergência mais rápida, embora exigindo controle rigoroso de hiper parâmetros como *max_depth* para evitar sobre ajuste (*overfitting*);

No contexto do *LightGBM*, o GOSS é essencial para balancear velocidade de treinamento e manutenção da representatividade estatística do conjunto de dados. Em métodos como *AdaBoost*, os pesos das amostras refletem diretamente sua importância. Porém, em modelos baseados em gradiente (como GBDT), essa informação precisa ser extraída indiretamente e os gradientes passam a desempenhar esse papel.

A intuição por trás do GOSS é dada por : Instâncias com pequenos gradientes indicam que o modelo já as aprendeu bem; Instâncias com grandes gradientes representam erros maiores, devendo receber maior atenção no treinamento.

Uma abordagem ingênua sugeriria descartar exemplos com gradientes baixos. Contudo, isso alteraria a distribuição original dos dados comprometendo a generalização do modelo. Para resolver isso, o GOSS mantém todas as instâncias com grandes gradientes e realiza amostragem aleatória nas de pequeno gradiente, aplicando a estas um fator de correção para preservar a distribuição original no cálculo do ganho de informação.

Algoritmo 1: Gradient-based One-Side Sampling (GOSS)

Passo	Descrição
Entrada	I: dados de treinamento, d: número de iterações
Entrada	a: proporção de amostragem dos dados com grande gradiente
Entrada	b: proporção de amostragem dos dados com pequeno gradiente
Entrada	*loss*: função de perda, L: modelo fraco (ex: árvore de decisão)
Inicialização	*models* ← {}, fact ← (1−a)/b
Amostragem	topN ← a × len(I), randN ← b × len(I)
Iteração	Para i = 1 até d, faça:
1	*preds* ← *models.predict*(I): calcula as predições dos modelos atuais
2	g ← *loss*(I, preds): calcula os gradientes da função de perda
3	w ← {1,1,...}: inicializa os pesos dos exemplos
4	*sorted* ← *GetSortedIndices*(abs(g)): ordena os índices pelo valor absoluto do gradiente
5	*topSet* ← *sorted*[1:topN]: seleciona os índices dos maiores gradientes
6	randSet ← *RandomPick*(*sorted*[topN:len(I)], randN): seleciona aleatoriamente exemplos de pequeno gradiente
7	*usedSet* ← topSet + randSet: une os conjuntos selecionados
8	w[randSet] ×= *fact*: ajusta o peso dos exemplos de pequeno gradiente
9	*newModel* ← L(I[*usedSet*], −g[*usedSet*], w[*usedSet*]): treina novo modelo fraco com os exemplos selecionados
10	*models.append*(*newModel*): adiciona o novo modelo à lista de modelos

Fonte: (Ke *et* al.; 2017).

Essa estratégia garante que o modelo se concentre nas instâncias mal compreendidas (com erro alto), sem comprometer a integridade da amostragem. No contexto da presente monografia, a utilização do *LightGBM* com GOSS mostrou-se extremamente eficaz para modelar variáveis agrícolas com alta variabilidade, como TCH (Toneladas de Cana por Hectare), em cenários complexos e heterogêneos.

2.7 VALIDAÇÃO DE MODELOS

A seleção de um modelo de aprendizado de máquina com desempenho robusto e generalizável exige um *framework* de avaliação metodologicamente rigoroso. Não basta treinar um algoritmo, é necessário validar sua capacidade preditiva em cenários que simulem sua aplicação em dados futuros e desconhecidos. Esta seção detalha os procedimentos e critérios adotados para garantir a validade estatística da comparação entre algoritmos e a subsequente seleção dos modelos mais performáticos para a predição de TCH (WESTPHAL; BRANNATH, 2019).

A seleção do algoritmo de regressão mais adequado é intrinsecamente ligada às características dos dados. Dados agrícolas, como os utilizados neste estudo, apresentam complexidades que demandam métodos de avaliação robustos. Entre as principais características, destacam-se: relações não-lineares complexas entre variáveis agronômicas, interações multifatoriais entre solo, clima e manejo, presença de *outliers* decorrentes de eventos climáticos extremos ou erros de medição, heterogeneidade espacial entre talhões e variabilidade temporal entre safras. Essas características tornam métodos que avaliam o desempenho de forma mais completa, como os *ensemble*s, particularmente adequados para problemas de predição agrícola (OLIVEIRA *et* al., 2021).

2.7.1 VALIDAÇÃO CRUZADA *K-FOLD CROSS-VALIDATION*

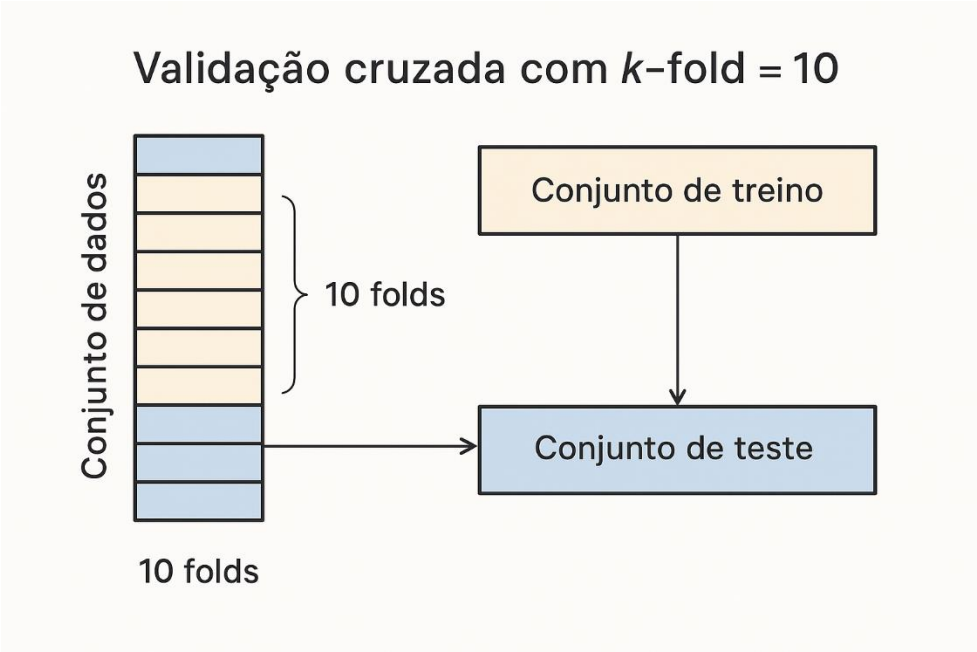
Ao treinar um modelo de *machine learning*, um dos maiores desafios é garantir que sua performance seja generalizável para dados futuros. Uma abordagem simples seria dividir os dados uma única vez em treino e teste, mas essa estratégia apresenta uma fraqueza significativa: o resultado pode variar drasticamente dependendo da divisão aleatória, gerando otimismo irreal sobre o desempenho.

Para superar essa limitação, este trabalho empregou a validação cruzada *k-fold*, considerada padrão-ouro para avaliação de modelos (HASTIE; TIBSHIRANI; FRIEDMAN, 2009). O processo, implementado através da função *setup()* do PyCaret com *fold*=10, funciona da seguinte forma: O particionamento de dados é realizado a partir do conjunto de dados dado. Este é dividido em $k = 10$ *folds* (partições) de tamanho aproximadamente igual. Então o processo iterativo é iniciado. Para cada *fold,* o modelo é treinado nos outros 9 *folds* e validado no *fold* restante. Quando finalizada esta etapa. A próxima etapa do método de é avaliação do modelo. Para cada tarefa específica de Aprendizado de Máquina, um tipo de métricas de

performance deve ser escolhida para realização do cálculo para cada uma das 10 iterações. A performance final do modelo é a média dos 10 resultados individuais (HASTIE; TIBSHIRANI; FRIEDMAN, 2009).

Esta abordagem garante que cada subconjunto de dados seja utilizado tanto para treino quanto para validação. Para tarefas de classificação as métricas comumente adotadas são: Acurácia, Precisão, Revogação e *F1-Score*. Já para a tarefa de Regressão as métricas mais comuns de avaliação são: Média Absoluta de Erro (MAE), Média Quadrada de Erro (MSE), Raiz da média Absoluta de Erro (RMAE) e Raiz Média Quadrada de Erro (RMSE) sendo essa última escolhida para detecção de resultados com bastante variância e anomalias (MARIO FILHO, 2018). O processo do *k-fold Cross-Validation* é ilustrado esquematicamente na Figura 1, na parte de robustez estatística.

Figura 1 - Representação esquemática do processo de validação cruzada com k=10.



Fonte: Elaborada pelo autor.

2.7.2 VALIDAÇÃO *HOLD-OUT*

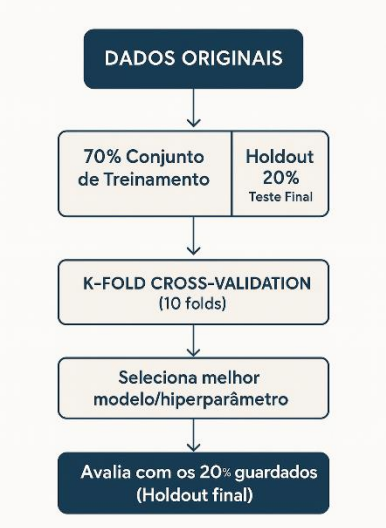
Após a seleção do modelo de melhor desempenho na validação cruzada, é fundamental avaliar sua capacidade de generalização em dados completamente novos, que não participaram de nenhuma etapa do treinamento ou ajuste de hiper parâmetros. Para isso, empregou-se o método de validação *Hold-out*, considerado padrão em estudos aplicados de aprendizado de máquina (HASTIE; TIBSHIRANI; FRIEDMAN, 2009).

No procedimento *Hold-out*, o conjunto de dados original é dividido em duas partes de forma estratificada: uma fração destinada ao treinamento e validação cruzada dos modelos (neste estudo, 70% dos dados) e uma fração reservada exclusivamente para teste final (os 30% restantes). O conjunto de teste permanece totalmente isolado durante todo o processo de seleção e ajuste dos modelos, sendo utilizado apenas na etapa final para estimar o desempenho preditivo realista do modelo escolhido (WESTPHAL; BRANNATH, 2019).

Após a seleção, o *Huber Regressor* foi treinado integralmente nos 70% dos dados de treino e, em seguida, avaliado no conjunto de teste (*Hold-out*, 30%). Essa abordagem permite aferir a robustez e a capacidade de generalização do modelo, simulando sua aplicação prática em cenários reais, onde as previsões são feitas sobre dados inéditos.

A Figura 2 ilustra esquematicamente o processo integrado de validação cruzada *k-fold* e validação *Hold-out*, destacando a separação dos conjuntos de treino e teste e o fluxo de avaliação dos modelos.

Figura 2 - Representação esquemática do processo de validação cruzada *k-fold* e validação *Hold-out.*



Fonte: (WESTPHAL; BRANNATH, 2019).

As métricas de desempenho calculadas no conjunto *Hold-out* como RMSE, MAE fornecem melhor a estimativa da performance do modelo em produção, já que refletem sua capacidade de prever corretamente a produtividade (TCH) em talhões nunca vistos anteriormente pelo algoritmo (HASTIE; TIBSHIRANI; FRIEDMAN, 2009; WESTPHAL; BRANNATH, 2019).

Essa metodologia rigorosa de avaliação, combinando validação cruzada para seleção e *Hold-out* para aferição final, garante a confiabilidade estatística dos resultados e a aplicabilidade prática do modelo desenvolvido.

2.8 MÉTRICAS DE AVALIAÇÃO

Para comparar quantitativamente o desempenho dos algoritmos, é necessário selecionar uma métrica alinhada aos objetivos do problema. Este trabalho adotou duas métricas dadas: o Erro Médio Absoluto (MAE), e a Raiz do Erro Médio Quadrado (RMSE). O MAE é definido na Equação 7 dada:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{7}$$

No qual n é o número total de observações (talhões), $\hat{y}_i$ é o valor de TCH predito pelo modelo para a observação y_i é o valor de TCH real para a observação i.

O RMSE é apresentado como segundo critério na Equação 8:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}}, \tag{8}$$

no qual as variáveis utilizadas são as mesmas já apresentadas na Equação 8.

A característica fundamental do RMSE reside no termo $(\hat{y}_i - y_i)^2$. Ao elevar o erro ao quadrado, a métrica não apenas penaliza erros maiores, mas o faz de forma exponencial. Isso significa que um único talhão cuja produtividade foi prevista com um erro de 20 t/ha contribui

para o RMSE total mais do que quatro talhões com erros de 10 t/ha cada. No contexto do planejamento agroindustrial, essa sensibilidade é uma vantagem estratégica. A subestimação drástica de um talhão de alta produtividade pode levar à perda de matéria-prima por falta de capacidade logística no pico da colheita. Inversamente, a superestimação pode resultar em ociosidade de frentes de colheita e caminhões, um custo operacional direto.

Ao contrário do Erro Absoluto Médio (MAE), que trata todos os erros de forma linear e igualitária, o RMSE força o algoritmo a dar prioridade máxima à redução desses desvios extremos, que são os mais disruptivos para a operação. Portanto, ao otimizar o modelo para minimizar o RMSE, estamos implicitamente alinhando o objetivo estatístico do modelo ao objetivo de negócio: mitigar os riscos associados às previsões mais imprecisas e aumentar a confiabilidade do planejamento de safra (CHAI; DRAXLER, 2014).

3 MATERIAIS E MÉTODOS

No presente capítulo apresenta o método o desenvolvimento e metodologia adotadas, abrangendo todas as etapas necessárias para o desenvolvimento, avaliação e preditivos da produtividade agrícola feitos pelo *framework*, AutoML PyCaret. Ao longo deste capítulo, são apresentados os conjuntos de dados, pré-processamento, a configuração dos algoritmos, as métricas de avaliação, assegurando a validação dos modelos estudados.

3.1 CONJUNTO DE DADOS

O conjunto de dados utilizado neste estudo é proveniente de registros operacionais e agronômicos de uma usina sucroenergética brasileira, representando um histórico produtivo real em nível de talhão. Os dados foram extraídos de sistemas de gestão agrícola (ERP Agrícola) e submetidos a um processo de seleção de características *(feature selection)*, resultando em um arquivo estruturado (base_modelo.csv).

O recorte final dos dados considerados para compor o *dataset* foi definido contendo 9.147 registros únicos, no qual cada registro corresponde a um talhão colhido em uma determinada safra, e 19 variáveis (dimensões). O *dataset* abrange um período de múltiplas safras (2018-2023), englobando algumas condições que para a construção de um modelo, tais como: variabilidade Genética, múltiplas variedades de cana-de-açúcar; variabilidade edáfica, diferentes tipos de solo (Latossolo, Argissolo); variabilidade de manejo, distintos estágios de corte e sistemas de irrigação e; variabilidade climática, registros de estações meteorológicas locais. Esta heterogeneidade multifatorial assegura a capacidade de generalização dos modelos treinados, condição essencial para aplicabilidade operacional (JAMES *et* al., 2021).

As dimensões selecionadas que compõem o conjunto de dados são apresentadas na Tabela 3, dada por:

Tabela 3 - Dimensões selecionadas para a modelagem de TCH

Categoria	Nome da Variável	Descrição	Tipo	Unidade
Variável Alvo	QT_TCHREALIZADO	Toneladas de Cana por Hectare (produtividade real)	Numérica	t/ha
Manejo	NO_CORTE	Idade do canavial expressa na ordem do corte	Numérica	-
Manejo	CD_SISTEMAIRRIGACAO	Sistema de irrigação (0: Sequeiro, 1: Irrigado)	Categórica	-
Qualidade	QT_ATRREALIZADO	Açúcar Total Recuperável, medido em kg por tonelada	Numérica	kg/t
Climática	VL_PRECIPITACAOANO	Precipitação anual acumulada no ciclo da cultura	Numérica	mm
Climática	Temp_Media	Temperatura média no ciclo da cultura	Numérica	°C
Logística	QT_KMDISTANCIA	Distância do talhão ao pátio da usina	Numérica	km
Física	QT_AREATOTAL	Área produtiva do talhão	Numérica	hectares
Edáfica	CD_TIPOSOLO	Código que identifica a classe de solo	Categórica	-
Genética	CD_VARIEDADE	Código que identifica a variedade genética da cana	Categórica	-

Fonte: Elaborado pelo autor (2025).

3.2 CONFIGURAÇÕES DO FRAMEWORK

Este estudo caracteriza-se como pesquisa aplicada de natureza quantitativa, com abordagem empírico-analítica, voltada ao desenvolvimento e validação de modelos preditivos de produtividade agrícola (TCH) utilizando técnicas de AutoML via PyCaret. Adota-se o paradigma hipotético-dedutivo (POPPER, 1959), no qual uma hipótese central é submetida a testes empíricos rigorosos através de experimentação controlada e análise estatística.

Para a configuração do framework foi necessário preencher vários parâmetros para a AutoML realize a identificação de modelos preditivos de TCH com desempenho melhor na métricas de avaliação RMSE, na qual RMSE deve ser inferior 8,5 t/ha, valor estimado em medições anteriores (baseline)

Após essa etapa é realizada uma seleção de características (dimensões) com base no conhecimento agronômico e análise de correlação. JOHNSON; WICHERN (2018). Na etapa seguinte, é realizado o pré-processamento, adotado foi a transformação de Yeo-Johnson, apresentado na Seção 2.4, para tratar valores ausentes e remoção de *outliers* extremos, definidos pelo critério do Intervalo Interquartil (IQR) TUKEY (1977). A aplicação da transformação de Yeo-Johnson, utilizada para normalizar distribuições não-gaussianas.

Após a configuração do framework, o motor AutoML realiza a etapa de treinamento, que nesse caso particular é executar a comparação sistemática de vários algoritmos de regressão utilizando a linha de código *compare_models*(), (HE *et* al.; 2021). Por fim a avaliação é apresentada para validar os modelos comparados (ABDI *et* al.; 2007).

Após todos os parâmetros serem definido, eles são inseridos e executados na função *setup()* do *framework* PyCaret. Nessa etapa serão executados todos os algoritmos disponíveis na base do PyCaret para a tarefa de regressão e os melhores serão apresentados com seus respectivos resultados.

O trecho de código abaixo detalha a configuração da função *setup()*.

```
s = setup(
    data=df,
    target='QT_TCHREALIZADO',
    session_id=42,
    normalize=True,
    transformation=True,
    remove_multicollinearity=True,
    multicollinearity_threshold=0.9,
    fold=10,
    verbose=False
)
```

Na Tabela 4, são apresentadas de forma detalhada os principais parâmetros de configuração.

Tabela 4 - Parâmetros Completos da Função setup() do PyCaret

Parâmetro	Valor/Config	Justificativa Técnica
data=df	DataFrame principal	**Base de dados** com 9.147 registros para treinamento e validação
target='QT_TCHREALIZADO'	Variável alvo	**Produtividade de TCH** (t/ha) como variável dependente a ser predita
session_id=42	Semente fixa	**Garantia de reprodutibilidade** em processos estocásticos e particionamento
normalize=True	Ativado	**Padronização Z-score** para algoritmos sensíveis à escala (regressões, SVR, KNN)
transformation=True	Ativado	**Transformação Yeo-Johnson** para normalizar distribuições assimétricas
remove_multicollinearity=True	Ativado	**Remoção automática** de variáveis com correlação > 0.9 para estabilidade do modelo
multicollinearity_threshold=0.9	Limiar de 0.9	**Critério de correlação** de Pearson para identificar variáveis redundantes
fold=10	10 partições	**Validação cruzada robusta** (k-fold) para estimativa confiável de generalização
verbose=False	Desativado	**Supressão de logs** detalhados para execução limpa em ambiente de produção

Fonte: Elaborado pelo autor (2025).

3.3 AVALIAÇÃO DE MODELOS

A avaliação da performance e a validação da capacidade de generalização dos modelos seguiram um protocolo rigoroso de duas fases, desenhado para garantir uma estimativa realista e não enviesada do desempenho.

Na etapa de Seleção (Via Validação Cruzada): Na etapa de comparação de modelos, a performance de cada algoritmo foi estimada utilizando um procedimento de validação cruzada estratificada com k=10. Este método consiste em dividir os dados de treino em 10 partições,

treinando e avaliando cada modelo 10 vezes, de modo que cada partição seja utilizada como conjunto de validação uma única vez. A performance final de cada modelo nesta fase foi a média dos resultados das 10 iterações. O RMSE foi a métrica primária utilizada para ordenar e selecionar o algoritmo mais promissor, dada sua sensibilidade a erros de grande magnitude. A aplicação da metodologia de validação cruzada *k-fold* e a classificação dos algoritmos com base na métrica RMSE culminaram na identificação de dois modelos com desempenho superior para a predição de TCH: o *Huber Regressor* e o *Light Gradient Boosting Machine* (*LightGBM*).

Na validação final, via Conjunto *Hold-out*, após a seleção do modelo que apresentaram melhor desempenho (*Huber Regressor*), foi treinado uma última vez utilizando todo o conjunto de dados de treino (70%). Sua performance final e definitiva foi, então, avaliada no conjunto de teste (*Hold-out*), correspondente aos 30% dos dados que foram mantidos isolados durante todo o processo de treino e seleção. Esta etapa fornece a estimativa mais fidedigna do desempenho do modelo em dados completamente novos.

Embora o RMSE tenha sido o critério de decisão, um conjunto de métricas foi calculado em ambas as fases para fornecer uma visão do comportamento do modelo. As principais métricas reportadas foram: RMSE, métrica primária de seleção dos algoritmos, que expressa o desvio padrão dos resíduos (erros de predição) e penaliza erros maiores de forma quadrática e; MAE, média do erro na mesma unidade da variável alvo (t/ha), facilitando a interpretação operacional.

O estudo foi conduzido utilizando um computador com a seguinte configuração: Processador: Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz 2.70 GHz; RAM instalada: 16,0 GB; Placa de vídeo Intel(R) HD Graphics 620 (128 MB); Sistema operacional Windows 11.

A linguagem utilizada foi Python (v. 3.11.11), com as seguintes bibliotecas: PyCaret (v. 3.3.2), *Framework* AutoML para execução dos experimentos de regressão; Scikit-learn (v. 1.3.0), Pandas (v. 2.0.3); Numpy (v. 1.2.43); LightGBM (v. 4.0.0); Matplotlib; e Seaborn. Os resultados gerados são apresentados e discutidos no Capítulo 4 – Resultados e Discussão.

4 RESULTADOS E DISCUSSÃO

No presente capítulo é apresentado os resultados obtidos a partir da aplicação da metodologia de AutoML. O foco da análise é a comparação sistemática dos algoritmos de regressão, a avaliação aprofundada do modelo *Huber Regressor* e a discussão das implicações práticas dos achados para a predição de produtividade da cana-de-açúcar (TCH).

A presença de um modelo linear robusto *(Huber Regressor*) e de um algoritmo de *ensemble* não linear (*LightGBM*) entre os melhores resultados evidencia a importância de se testar diferentes famílias de modelos para problemas complexos como a predição de TCH. Enquanto o *Huber Regressor* se destacou pela capacidade de lidar com *outliers* e fornecer previsões estáveis em cenários com dados ruidosos, o *LightGBM* mostrou-se superior na modelagem de relações não lineares e interações entre múltiplas variáveis agronômicas. Essa diversidade reforça a relevância de ferramentas de AutoML, como o PyCaret, que facilitam a experimentação sistemática e a comparação entre algoritmos.

4.1 ANÁLISE COMPARATIVA E SELEÇÃO DO MODELO

A seleção do modelo foi realizada por meio de um processo competitivo automatizado, utilizando a função *compare_models()* do PyCaret. Mais de 15 algoritmos de regressão foram treinados e avaliados sob protocolo de validação cruzada estratificada (*10-fold*). A Tabela 8 apresenta os resultados de desempenho dos dois principais modelos selecionados para este estudo, ordenados pela métrica primária de avaliação, RMSE.

Tabela 5 - Desempenho Comparativo dos Modelos de Regressão (Validação Cruzada)

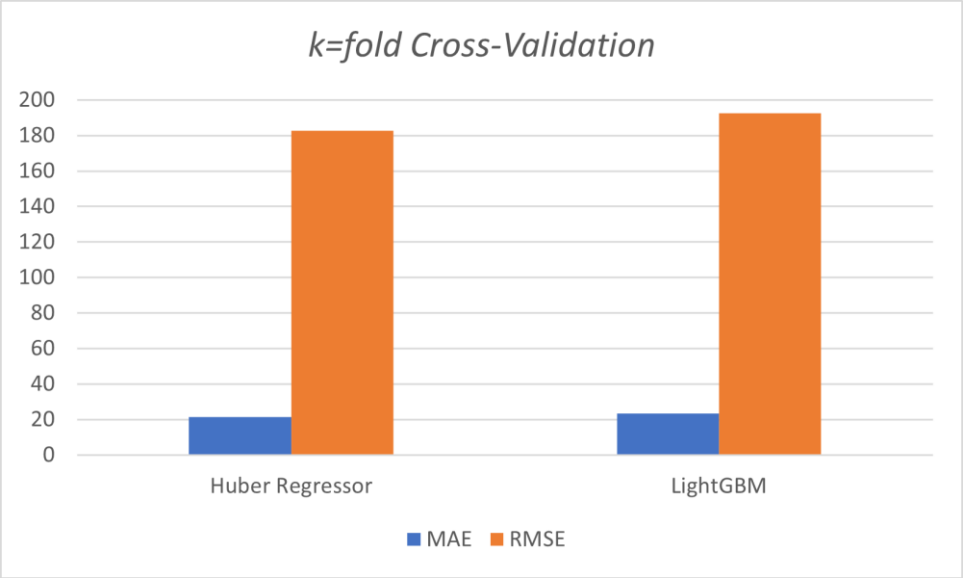
Modelo	MAE (t/ha)	RMSE (t/ha)	Tempo (s)
Huber Regressor	**21.35**	**182.67**	**9.03**
Light Gradient Boosting Machine	23.46	192.54	50.25

Fonte: O autor (2025), a partir de tabela_comparacao_final.csv

A análise da tabela demonstra que o *Huber Regressor* alcançou o menor RMSE médio (182,67 t/ha), estabelecendo-se como o modelo de melhor desempenho sob o critério primário. O *LightGBM*, embora seja um ensemble de última geração (KE *et* al., 2017), apresentou RMSE

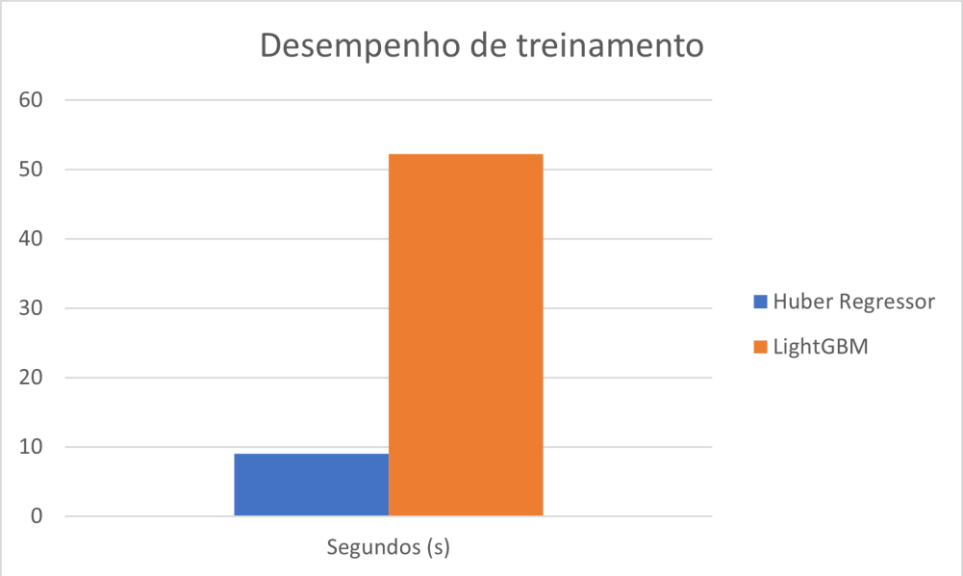
superior (192,54 t/ha), como apresentado no Gráfico 1.

Gráfico 1 - Comparativo de Erros entre as técnicas *Huber Regressor* e LightGBM



Fonte: Elaborada pelo Autor

Gráfico 2 - Comparativo de desempenho de treinamento entre *Huber Regressor* e *LightGBM*



Fonte: Elaborada pelo autor

O gráfico 2 reforçou a superioridade do *Huber Regressor* na métrica de seleção RMSE. Sua robustez a *outliers*, característica central do algoritmo (HUBER, 1964), foi determinante para lidar com a alta variabilidade dos dados agrícolas. Além disso, o tempo de treinamento do Huber foi significativamente menor, um diferencial relevante para aplicações práticas e

retreinamentos frequentes.

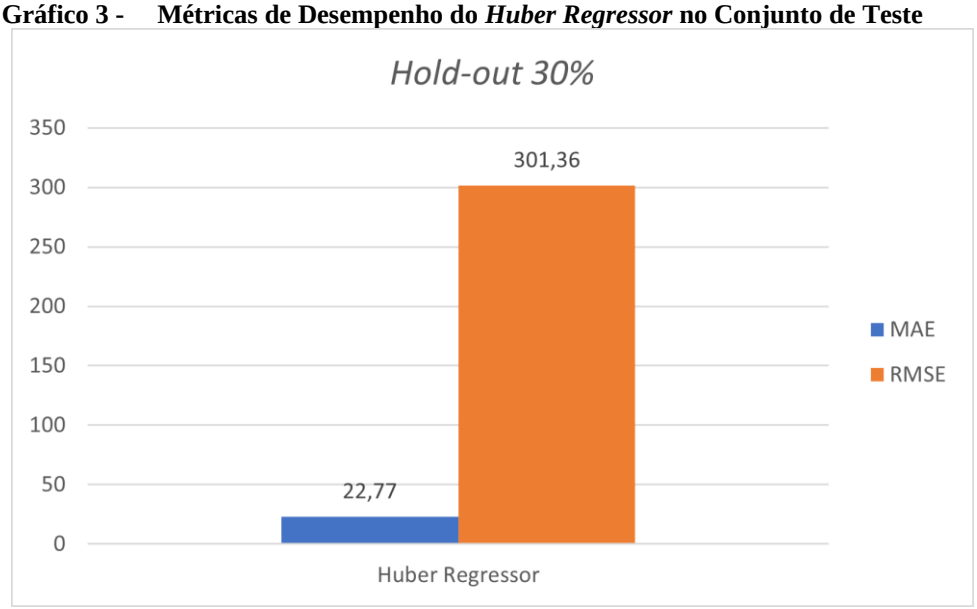
Após a seleção, o *Huber Regressor* foi treinado no conjunto de treino (70%) e avaliado no conjunto de teste (*Hold-out*, 30%). Os resultados estão na Tabela 6 e Gráfico 3.

Tabela 6 - Métricas de Desempenho do *Huber Regressor* no Conjunto de Teste

Métrica	Valor	Unidade
MAE	22,77	t/ha
RMSE	301,36	t/ha

Fonte: Elaborada pelo Autor.

Gráfico 3 - Métricas de Desempenho do *Huber Regressor* no Conjunto de Teste



Fonte: Elaborada pelo Autor.

A avaliação no conjunto de teste revela a real dificuldade do problema: o MAE de 22,77 t/ha representa o erro médio operacional, um valor elevado diante da média de produtividade (~75 t/ha). O RMSE de 301,36 t/ha, muito superior ao MAE, confirma que a validação *Hold-out* acentua a presença de *outliers* nos resultados preditos.

4.2 DISCUSSÃO GERAL E IMPLICAÇÕES PRÁTICAS

Os resultados deste estudo, em conjunto, contam uma história coesa. A superioridade do *Huber Regressor* sobre *ensembles* mais complexos, como *LightGBM*, sugere que a principal característica para modelar os dados deste problema não foi a capacidade de capturar interações complexas, mas sim a robustez para lidar com a alta variabilidade e os *outliers* inerentes aos dados agrícolas (HUBER, 1964).

A escolha do *Huber Regressor* se justifica por critérios práticos e metodológicos observados ao longo do estudo. Em primeiro lugar, ao comparar diretamente os desempenhos, o *Huber Regressor* apresentou resultados mais estáveis e menos sensíveis a oscilações extremas dos dados, enquanto o *LightGBM*, embora sofisticado e capaz de modelar relações não-lineares, mostrou-se mais suscetível à influência de *outliers* presentes na base. Essa diferença foi perceptível tanto nas métricas de erro quanto na consistência das previsões em conjuntos de validação e teste.

O *LightGBM* é reconhecido por sua eficiência e capacidade de capturar padrões complexos em grandes volumes de dados, sendo frequentemente a escolha padrão em competições de aprendizado de máquina. No entanto, neste contexto específico, sua sensibilidade a valores extremos resultou em maior variabilidade dos resultados, o que pode comprometer a confiabilidade das previsões em cenários reais, onde anomalias e desvios são inerentes ao processo produtivo agrícola. Já o *Huber Regressor*, ao adotar uma função de perda que penaliza de forma quadrática os erros pequenos e de forma linear os erros grandes, reduzindo o impacto dos *outliers* sem descartar sua informação. Isso garante que o modelo permaneça centrado no comportamento predominante dos dados, favorecendo a robustez e a generalização.

Além disso, a implementação via *framework low-code* PyCaret, destacou a importância de soluções que conciliem facilidade de uso, replicabilidade e estabilidade dos resultados. O *Huber Regressor* demonstrou ser não apenas eficaz, mas também transparente e facilmente interpretável, facilitando a comunicação dos resultados para equipes multidisciplinares e para a tomada de decisão operacional.

Portanto, a escolha do *Huber Regressor* mostrou-se mais adequada para o problema apresentado para o *framework*, no qual a presença de variabilidade e eventos extremos foram presentes, mas tratadas de modo que não comprometeram as previsões.

5 CONCLUSÃO

Este trabalho teve como objetivo avaliar o potencial do Aprendizado de Máquina Automatizado (AutoML), com ênfase no uso do *framework* PyCaret, para a predição da produtividade agrícola (TCH) em usinas de cana-de-açúcar. A hipótese central era que o AutoML permitiria identificar modelos preditivos robustos, capazes de superar abordagens manuais em cenários de dados agrícolas complexos.

Os resultados demonstraram que, entre vários algoritmos avaliados, o *Huber Regressor* apresentou os menores erros nos critérios de avaliação selecionados, superando outros modelos como o *LightGBM*. O desempenho do *Huber Regressor* manteve-se estável testes realizados, apresentando a presença de *outliers* sobre as previsões do TCH.

A principal limitação refere-se à dependência da abstração proporcionada pelo PyCaret, que, por ser uma ferramenta low-code, automatiza etapas essenciais do pipeline de machine learning, como pré-processamento, seleção e ajuste de hiper parâmetros. Embora essa automação facilite o uso e acelere o desenvolvimento de modelos, pode restringir o controle detalhado do pesquisador sobre decisões técnicas específicas, limitando a customização avançada dos experimentos como a questão da interpretação e transparência dos modelos. A automação promovida pelo PyCaret dificultou a compreensão aprofundada das etapas internas do fluxo de funcionamento, especialmente no que diz respeito à engenharia de atributos, tratamento de *outliers* e seleção automática de variáveis. Isso pode impactar a interpretação dos resultados e a explicação das decisões do modelo para usuários finais e especialistas do domínio

Embora o PyCaret ofereça métricas padrão para avaliação de modelos, a customização de métricas específicas ou a implementação de estratégias de avaliação não convencionais pode ser restrita, o que limita a análise em cenários que exigem critérios diferenciados de performance reforçando a necessidade de avaliações críticas ao adotar *frameworks low-code* em projetos científicos, especialmente quando o objetivo é garantir máxima flexibilidade, transparência e controle sobre o desenvolvimento de modelos preditivos.

Como trabalhos futuros testes com abordagens híbridas e redes profundas, tais como: redes neurais e modelos híbridos, são planejados, com o objetivo de estudar o comportamento de relações não lineares e melhorar interações complexas.

REFERÊNCIAS

ABDI, H. ***The Bonferroni and Šidák Corrections for Multiple Comparisons***. In: Salkind, N. (ed.). Encyclopedia of Measurement and Statistics. Thousand Oaks: Sage, 2007.

ALI, M. ***PyCaret: an open-source, low-code machine learning library in Python***. Journal of Open Source Software, v. 5, n. 54, p. 2538, 2020.

BREIMAN, L. ***Bagging predictors***. Machine learning, v. 24, n. 2, p. 123-140, 1996. DOI: 10.1007/BF00058655.

BREIMAN, L. ***Random forests***. Machine learning, v. 45, n. 1, p. 5-32, 2001. DOI: 10.1023/A:1010933404324.

CHEN, T.; GUESTRIN, C. ***XGBoost: a scalable tree boosting system***. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. San Francisco: ACM, 2016. p. 785-794.

COMPANHIA NACIONAL DE ABASTECIMENTO (CONAB). **Acompanhamento da safra brasileira de cana-de-açúcar: safra 2024/25**. Brasília: CONAB, 2024. Disponível em:< https://www.gov.br/conab/pt-br/assuntos/noticias/safra-2024-25-de-cana-de-acucar-encerra-com-producao-estimada-em-676-96-milhoes-de-toneladas>. Acesso em: 23 jun. 2025.

DOORENBOS, J.; KASSAM, A. H. **Yield response to water**. FAO Irrigation and Drainage Paper 33. Rome: FAO, 1979.

FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P.; UTHURUSAMY, R. ***From data mining to knowledge discovery in databases***. AI Magazine, v. 17, n. 3, p. 37-54, 1996.

FRIEDMAN, J. H. ***Greedy function approximation: a gradient boosting machine***. Annals of Statistics, v. 29, n. 5, p. 1189-1232, 2001.

GEURTS, P.; ERNST, D.; WEHENKEL, L. ***Extremely randomized trees***. Machine learning, v. 63, n. 1, p. 3-42, 2006. DOI: 10.1007/s10994-006-6226-1.

GREGO, C. R. **Tecnologias desenvolvidas em Agricultura de Precisão**. (ed.). Agricultura de Precisão: resultados de um novo olhar. Brasília: Embrapa, 2020. p. 167-186.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. ***The elements of statistical learning: data mining, inference, and prediction***. 2. ed. New York: Springer, 2009. 745 p.

HE, X.; ZHAO, K.; CHU, X. ***AutoML: a survey of the state-of-the-art***. Knowledge-Based Systems, v. 212, p. 106622, 2021. DOI: 10.1016/j.knosys.2020.106622.

HUBER, P. J. ***Robust estimation of a location parameter***. The Annals of Mathematical Statistics, v. 35, n. 1, p. 73-101, 1964.

KE, G. *et al*. ***LightGBM: a highly efficient gradient boosting decision tree***. Advances in Neural Information Processing Systems, v. 30, p. 3146-3154, 2017.

MARIO FILHO. **As Métricas Mais Populares para Avaliar Modelos de Machine Learning**. 2018. Disponível em: <https://mariofilho.com/as-metricas-mais-populares-para-avaliar-modelos-de-machine-learning/>. Acesso em: 2 jul. 2025.

OLIVEIRA, P. H. *et al*. **Aplicação de XGBoost na predição de produtividade de cana-de-açúcar**. Ciência Rural, v. 51, n. 7, p. e20200123, 2021.

OLIVEIRA, R. A. *et al*. ***Ratoon management in sugarcane: challenges and opportunities***. Field Crops Research, v. 270, 2020.

PAREKH, M. ***PyCaret: an open-source, low-code machine learning library in Python***. Disponível em: <https://pycaret.gitbook.io/docs>. Acesso em: 26 jun. 2025.

PEDREGOSA, F. *et al*. ***Scikit-learn: machine learning in Python***. Journal of Machine learning Research, v. 12, p. 2825-2830, 2011.

POPPER, K. ***The logic of scientific discovery***. London: Routledge, 1959.

PROKHORENKOVA, L. *et al*. ***CatBoost: unbiased boosting with categorical features***. Advances in Neural Information Processing Systems, v. 31, p. 6638-6648, 2018.

PYCARET. ***Easy MLOps with PyCaret and MLflow***. 2022. Disponível em: <https://pycaret.gitbook.io/docs/learn-pycaret/official-blog/easy-mlops-with-pycaret-and-mlflow>. Acesso em: 2 jul. 2025.

SANTOS, M. F. *et al*. **Support Vector Machines para estimativa de TCH em canaviais**. Engenharia Agrícola, v. 39, n. 4, p. 445-452, 2019.

SILVA, J. C. *et al*. **Predição de produtividade de cana-de-açúcar utilizando Random Forest**. Revista Brasileira de Engenharia Agrícola e Ambiental, v. 24, n. 8, p. 532-538, 2020.

SILVA, M. A. *et al*. **Relationships between physiological traits and productivity of sugarcane in response to water deficit**. Sugar Tech, v. 23, p. 1075-1084, 2021.

SCARPARI, Maximiliano Salles. **Modelos para a previsão da produtividade da cana-de-açúcar**. 2002. 160 f. Tese (Doutorado em Agronomia) – Escola Superior de Agricultura "Luiz de Queiroz", Universidade de São Paulo, Piracicaba, 2002. Disponível em: <https://teses.usp.br/teses/disponiveis/11/11136/tde-17122002-165859/pt-br.php>. Acesso em: 1 jul. 2025.

SNOEK, J.; LAROCHELLE, H.; ADAMS, R. P. ***Practical Bayesian optimization of machine learning algorithms***. In: Advances in Neural Information Processing Systems 25 (NIPS 2012). Lake Tahoe, NV, 2012. p. 2960-2968.

WESTPHAL, M.; BRANNATH, W. ***Evaluation of multiple prediction models: a novel view on model selection and performance assessment***. BMC Medical Research Methodology, v. 19, n. 1, 2019. Disponível em: <https://pmc.ncbi.nlm.nih.gov/articles/PMC7270727/>. Acesso em: 2 jul. 2025.

WILLMOTT, C. J.; MATSUURA, K. ***Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance***. Climate Research, v. 30, n. 1, p. 79-82, 2005. DOI: 10.3354/cr030079.

WHITE, F.; *et al.* ***AI in Agriculture: Opportunities, Challenges, and Recommendations***. Council for Agricultural Science and Technology (CAST), 2025. Disponível em: <https://cast-science.org/wp-content/uploads/2025/03/CAST_AI-in-Agriculture.pdf.> Acesso em: 2 jul. 2025.

WOLPERT, D. H. ***Stacked generalization***. Neural Networks, v. 5, n. 2, p. 241-259, 1992.

YEO, I. K.; JOHNSON, R. A. ***A new family of power transformations to improve normality or symmetry***. Biometrika, v. 87, n. 4, p. 954-959, 2000.